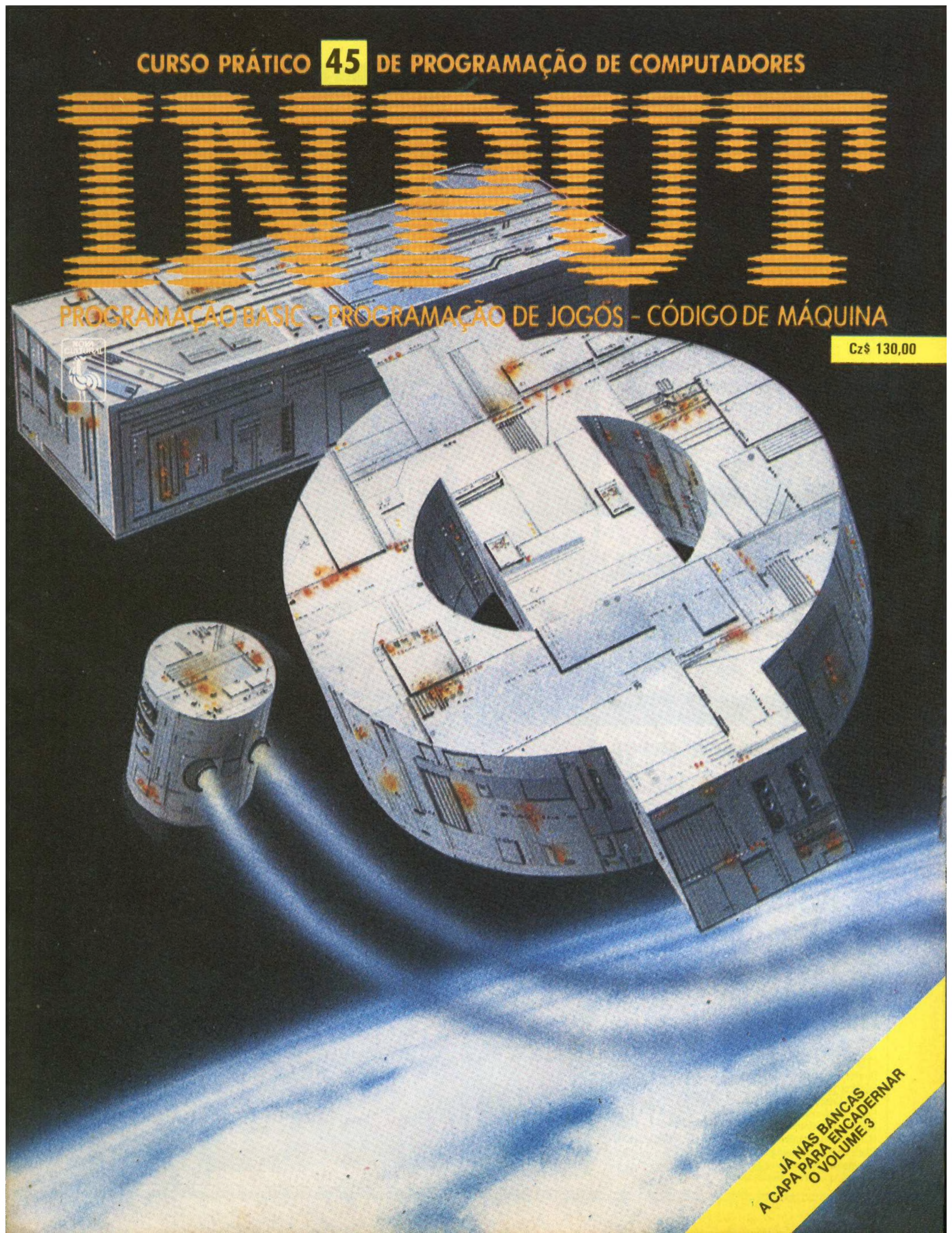
CURSO PRÁTICO 45 DE PROGRAMAÇÃO DE COMPUTADORES

INFO

PROGRAMAÇÃO BASIC - PROGRAMAÇÃO DE JOGOS - CÓDIGO DE MÁQUINA

Cz$ 130,00



JÁ NAS BANCAS
A CAPA PARA ENCADERNAR
O VOLUME 3

INPUT

Vol. 3 N.º 45

NESTE NÚMERO

PROGRAMAÇÃO DE JOGOS

O BANDIDO DE UM BRAÇO SÓ (2)

Numa poeirenta rua de uma cidade do velho oeste, você caminha lentamente para um duelo ao pôr-do-sol. Seu adversário só tem um braço, mas é mais rápido no gatilho do que Billy the Kid. Viva as emoções de um duelo como esse, jogando com um caça-níqueis 881

PROGRAMAÇÃO BASIC

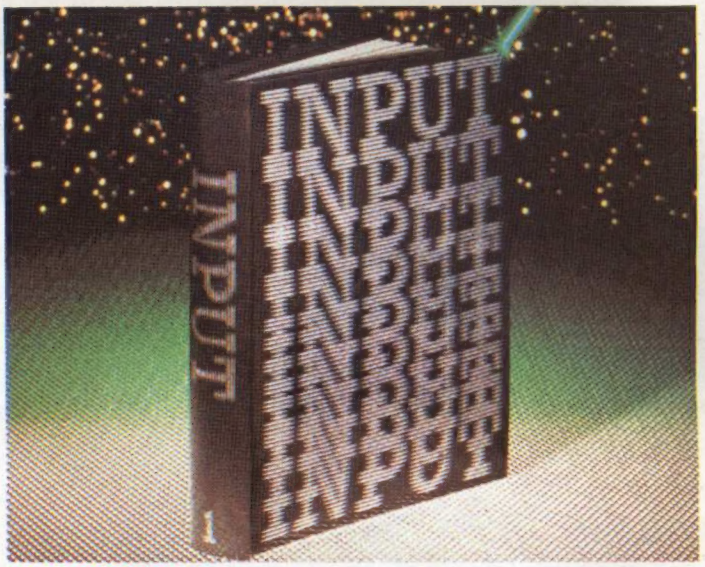
MENSAGENS SECRETAS

Agora, você está num país estranho, cercado de espiões de uma potência inimiga. Mas não fique desesperado: aprenda a criar seus próprios códigos secretos e entre para a história das mensagens cifradas 888

PROGRAMAÇÃO BASIC

ARMAZENAGEM DE NÚMEROS

Expoentes. Como são armazenados os números. Ponto flutuante e números negativos. A função **INSTR. PEEK** na memória. Dicas para economizar espaço na memória. Efeitos estranhos. Formatação com **PRINT USING** 894



PLANO DA OBRA
INPUT é uma obra editada em fascículos semanais, e cada conjunto de 15 fascículos compõe um volume. A capa para encadernação de cada volume estará à venda oportunamente.

FÉRIAS, VIAGENS, MUDANÇAS...
NÃO FIQUE COM A COLEÇÃO INCOMPLETA
Se você está saindo de férias, pretende viajar ou vai se ausentar por algum tempo, avise antecipadamente seu jornaleiro. Ele pode guardar os seus fascículos enquanto você estiver fora. Se, por qualquer motivo, você perdeu alguns números, peça-os também a seu jornaleiro, ou entre em contato com nossa Distribuidora:

1. **Pessoalmente** — Em *São Paulo*, os endereços são: Rua Brigadeiro Tobias, 773, Centro, telefone 227-4188; Av. Industrial, 117, Santo André, telefone 449-0411, das 7h30 às 17h00 - dias úteis. No *Rio de Janeiro*, Av. Mem de Sá, 191/193, Centro, telefone (021) 222-7422, das 7h30 às 17h00 - dias úteis.
2. **Por carta** — Envie para:
   DINAP — Distribuidora Nacional de Publicações
   Números Atrasados
   Estrada Velha de Osasco, 132 — Jardim Teresa
   CEP 06040 — Osasco — SP
3. **Por telex** — Utilize o nº (11) 33 670 DNAP.

Em **Portugal**, os pedidos devem ser feitos à Distribuidora Jardim de Publicações Lda. — Qta. Pau Varais, Azinhaga de Fetais, 2685, Camarate, Lisboa; Apartado 57; Telex 43 069 JARLIS P.
Atenção: Após seis meses do encerramento da coleção, o atendimento dos pedidos dependerá da disponibilidade do estoque.
Obs.: Quando pedir livros, mencione sempre o título e/ou o autor da obra, além do número da edição.

COLABORE CONOSCO
Encaminhe seus comentários, críticas, sugestões ou reclamações ao
SERVIÇO DE ATENDIMENTO AO LEITOR
Caixa Postal 9 442, São Paulo — SP.

EDITOR
RICHARD CIVITA
NOVA CULTURAL

Presidente
Flávio Barros Pinto
Diretoria
Carmo Chagas, Iara Rodrigues
Pierluigi Bracco, Plácido Nicoletto,
Roberto Silveira, Shozi Ikeda,
Sônia Carvalho

REDAÇÃO
Diretor Editorial: Carmo Chagas
Editores Executivos:
Stefania Crema, Berta Sztark Amar
Editor Chefe: Paulo de Almeida
Editoras Assistentes: Ana Lúcia B. de Lucena, Marisa Soares de Andrade
Chefe de Arte: Carlos Luiz Batista
Assistentes de Arte: Dagmar Bastos Sampaio, Grace Afonso Arruda, Mônica Lenarduni Corrau

Secretário de Redação: Mauro de Queiroz
Colaboradores
Consultor Editorial Responsável:
Dr. Renato M. E. Sabbatini
(Diretor do Núcleo de Informática Biomédica da Universidade Estadual de Campinas)
Execução Editorial: DATAQUEST Assessoria em Informática Ltda., Campinas - SP
Tradução, adaptação, programação e redação:
Abílio Pedro Neto, Aluísio J. Dornellas de Barros, Marcelo R. Pires Therezo, Marcos Huascar Velasco, Raul Neder Porrelli, Ricardo J. B. de Aquino Pereira.
Coordenação Geral: Rejane Felizatti Sabbatini

COMERCIAL
Diretor Comercial: Roberto Silveira
Gerente Comercial: Joaquim Celestino da Silva
Gerente de Circulação: Denise Mozol
Gerente de Propaganda e Publicidade: José Carlos Madio
Gerente de Pesquisa e Análise de Mercado: Wagner M. P. Nabuco de Araújo

CLC

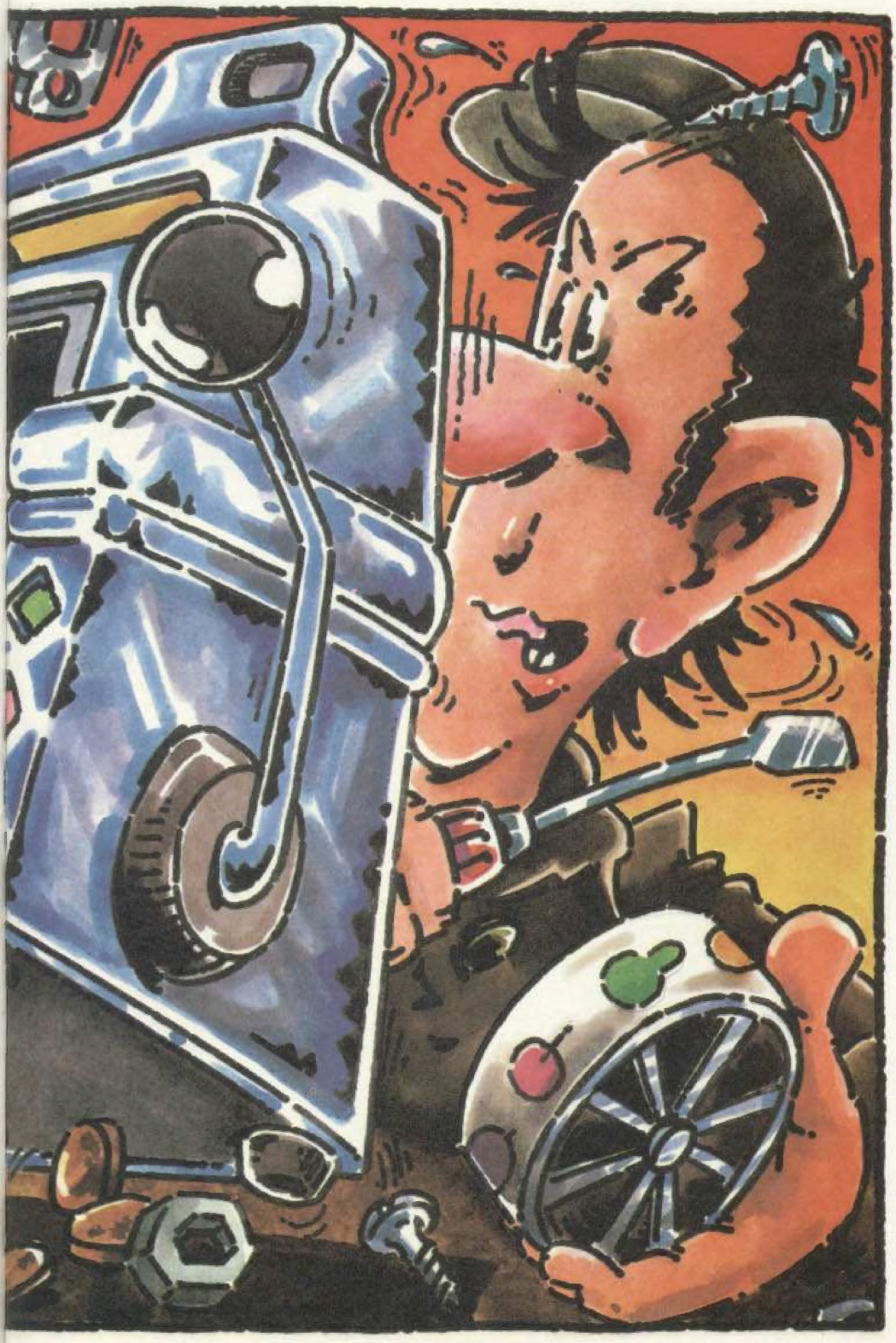
A Editora Nova Cultural Ltda. é uma empresa do Grupo CLC - Comunicações, Lazer, Cultura S.A.
Presidente: Richard Civita
Diretoria: Flávio Barros Pinto, João Gomez, Menahem M. Politi, Renê C. X. Santos, Stélio Alves Campos

© Marshall Cavendish Limited, 1984/85.
© Editora Nova Cultural Ltda., São Paulo, Brasil, 1986, 2ª edição, 1987.

Edição organizada pela Editora Nova Cultural Ltda.
Av. Brigadeiro Faria Lima, 2000 - 3º andar
CEP 01452 - São Paulo - SP - Brasil
(Artigo 15 da Lei 5 988, de 14/12/1973).
Esta obra foi composta pela AM Produções Gráficas Ltda. e impressa pela Companhia Lithographica Ypiranga.

O BANDIDO DE UM BRAÇO SÓ (2)

- COMPLETE SEU JOGO CAÇA-NÍQUEIS
- A ROTINA PRINCIPAL
- GIRE AS RODINHAS
- OS RESULTADOS PREMIADOS



Depois de completar o jogo com esta segunda parte, puxe a manivela e tente obter três frutas iguais no caça-níqueis. Esse resultado pagará vinte vezes a sua aposta.

Segunda e última parte do nosso jogo caça-níqueis, este artigo nos coloca em condições de executar todo o programa, cujo primeiro segmento você já deve ter gravado.

APERTEM OS CINTOS

```
210 LET HOLD=0
220 LET TOTAL=TOTAL-10: LET NU
DGE=0: PRINT AT 13,26; INK 2;"
    "
230 IF HFLAG=0 THEN LET HOLD=
0
240 FOR I=1 TO 3: FOR J=1 TO
12: SOUND .001,60
250 IF HOLD=0 THEN PRINT AT 7
,10;A$(J);AT 7,15;B$(J);AT 7,
20;C$(J);AT 10,10;A$(J+1);AT
10,15;B$(J+1);AT 10,20;C$(J+1)
;AT 13,10;A$(J+2);AT 13,15;B$(
J+2);AT 13,20;C$(J+2): NEXT J:
 NEXT I
270 IF HOLD=1 THEN PRINT AT 7
,15;B$(J);AT 7,20;C$(J);AT 10,
 15;B$(J+1);AT 10,20;C$(J+1);
AT 13,15;B$(J+2);AT 13,20;C$(J
+2): NEXT J: NEXT I
280 IF HOLD=6 THEN PRINT AT 7
,15;B$(J);AT 10,15;B$(J+1);AT
13,15;B$(J+2): NEXT J: NEXT I
290 IF HOLD=2 THEN PRINT AT 7
,10;A$(J);AT 7,20;C$(J);AT 10,
10;A$(J+1);AT 10,20;C$(J+1);AT
13,10;A$(J+2);AT 13,20;C$(J+2)
: NEXT J: NEXT I
300 IF HOLD=5 THEN PRINT AT 7
,10;A$(J);AT 10,10;A$(J+1);AT
13,10;A$(J+2): NEXT J: NEXT I
310 IF HOLD=3 THEN PRINT AT 7
,10;A$(J);AT 7,15;B$(J);AT 10,
10;A$(J+1);AT 10,15;B$(J+1);AT
 13,10;A$(J+2);AT 13,15;B$(J+2
): NEXT J: NEXT I
320 IF HOLD<>1 AND HOLD<>4 AND
HOLD<>6 THEN LET M=INT (RND*
12): IF M=0 THEN LET M=1
330 IF HOLD<>2 AND HOLD<>5 AND
```

```
HOLD<>4 THEN  LET K=INT (RND*
12): IF K=0 THEN  LET K=1
340 IF HOLD<>3 AND HOLD<>5 AND
HOLD<>6 THEN  LET L=INT (RND*
12): IF L=0 THEN  LET L=1
350 LET HOLD=0
360 PRINT AT 7,10;A$(M);AT 7,
15;B$(K);AT 7,20;C$(L);AT 10,
10;A$(M+1);AT 10,13;B$(K+1);AT
10,20;C$(L+1);AT 13,10;A$(M+2)
;AT 13,15;B$(K+2);AT 13,20;C$(
L+2)
```

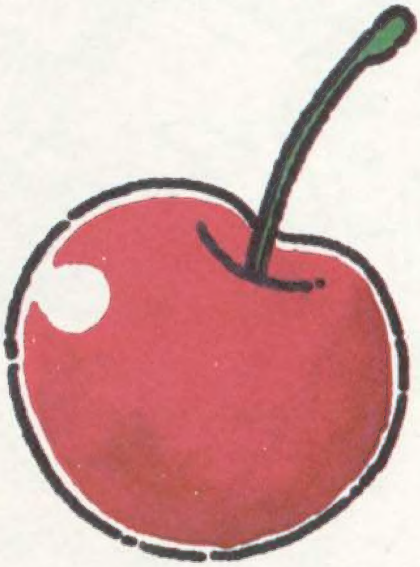
A linha 210 coloca zero na variável que segura uma ou mais rodas, que podem ser escolhidas através do teclado. A rotina verifica qual é o conteúdo dessa variável e faz com que se movimentem as rodas que estão livres.

Depois de girar as rodas, a rotina recoloca zero em **HOLD** na linha 350 — os botões que seguram as rodas são apagados — e, em seguida, a linha 360 incumbe-se de desenhá-las em sua posição final, paradas.

VERIFIQUE O RESULTADO

```
370 GOSUB 510
510 LET T$=A$(M)+B$(K)+C$(L)
520 LET L$=A$(M+1)+B$(K+1)+C$(
L+1)
530 LET L$=A$(M+2)+B$(K+2)+C$(
L+2)
540 GOSUB 660
550 RETURN
680 LET TEMP=TOTAL
690 IF M$( TO 4)=M$(5 TO 8)
AND M$( TO 4)=M$(9 TO ) THEN
LET TOTAL=TOTAL+50: IF M$( TO
4)=C$(4) THEN  LET TOTAL=TOTAL
+5000: GOTO 640
```

```
700 IF M$( TO 4)=C$(1) AND M$(
TO 4)=M$(5 TO 8) AND M$( TO 4)
=M$(9 TO ) THEN  LET TOTAL=TOT
AL+50
710 IF M$( TO 4)=C$(3) AND M$(
TO 4)=M$(5 TO 8) AND M$( TO 4)
=M$(9 TO ) THEN  LET TOTAL=TOT
AL+50
720 IF M$( TO 4)=A$(3) THEN
LET TOTAL=TOTAL+10: IF M$(5 TO
 8)=A$(3) THEN  LET TOTAL=TOTA
L+10
```



```
730 IF TOTAL>TEMP THEN  FOR I=
1 TO 6: SOUND .05,50: NEXT I
740 LET DD=INT (TOTAL/100):
LET CC=TOTAL-(DD*100): PRINT
INK 2; PAPER 6;AT 17,0;"$
 ";AT 18,0;"C      "; PAPER 7;
BRIGHT 1;AT 17,1;" ";DD;AT 18,
1;" ";CC
750 IF TOTAL<1 THEN  GOTO 760
760 RETURN
```

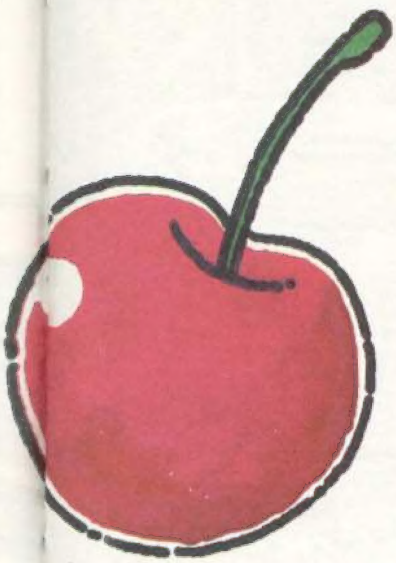
A linha 370 salta para a sub-rotina da linha 510, que coloca as três linhas mostradas pelas rodas nas variáveis alfanuméricas **T$**, **M$** e **L$**. A linha do meio, **M$**, é a que vale para fins de contagem de pontos.

A sub-rotina que faz isso começa na linha 680. Ela verifica se ocorreu, em algum lugar, uma linha premiada, e soma o valor do prêmio ao patrimônio acumulado pelo jogador.

COMO DAR UM EMPURRÃOZINHO

```
380 IF M<7 OR K=L OR L>2 THEN
LET NUDGE=1: PRINT  BRIGHT 1;
PAPER 7; INK 2;AT 13,26;"NUDGE
"
390 LET HFLAG=INT (RND+.5): IF
HFLAG=1 THEN  FOR I=1 TO 19
STEP 5: PRINT AT 16,I; INK 6;
BRIGHT 1;"HOLD";: NEXT I
400 IF INKEY$<>"" THEN  GOTO
400
410 LET I$=INKEY$: IF I$=""
THEN  GOTO 410
420 IF I$=" " THEN  FOR I=9 TO
19 STEP 5: PRINT  INK 2;AT 16,
I;"     ": NEXT I: GOTO 210
430 IF I$="E" AND NUDGE=1 THEN
 GOSUB 600: LET NUDGE=0: PRINT
AT 13,26; INK 2;"      ": SOUND
.1,30: GOSUB 510: LET RN=INT (
RND*10): IF INT (RN/2)=RN/2
AND HFLAG<>1 THEN  LET NUDGE=1
: PRINT AT 13,26; INK 7;
BRIGHT 1;"NUDGE": GOTO 400
440 IF I$="Q" AND NUDGE=1 THEN
 GOSUB 560: LET NUDGE=0: PRINT
AT 13,26; INK 2;"      ": SOUND
.1,30: GOSUB 510: LET RN=INT (
RND*10): IF INT (RN/2)=RN/2
AND RN<3 THEN  LET NUDGE=1:
PRINT AT 13,26; INK 7; BRIGHT
1;"NUDGE": GOTO 400
450 IF I$="W" AND NUDGE=1 THEN
 GOSUB 580: LET NUDGE=0: PRINT
AT 13,26; INK 2;"      ": SOUND
.1,30: GOSUB 510
```

```
460 IF I$="D" AND NUDGE=1 THEN
 GOSUB 620: LET NUDGE=0: PRINT
AT 13,26; INK 2;"      ": SOUND
.1,30: GOSUB 510: LET RN=INT (
RND*10): IF INT (RN/2)<>RN/2
THEN  LET NUDGE=1: PRINT AT 13
,26; INK 7; BRIGHT 1;"NUDGE":
```



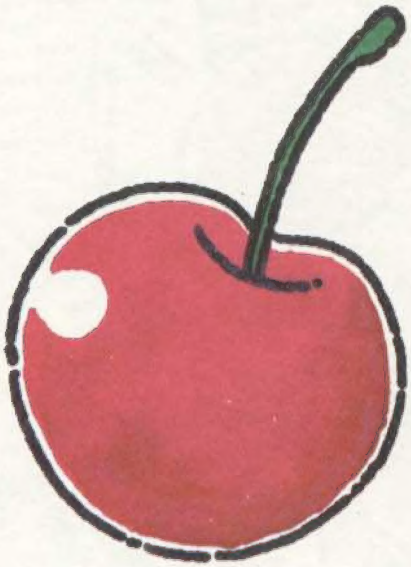
```
610 PRINT AT 7,20;C$(L);AT 10,
20;C$(L+1);AT 13,20;C$(L+2):
RETURN
620 LET L=L-1: IF L<1 THEN
LET L=L+12
630 PRINT AT 7,20;C$(L);AT 10,
20;C$(L+1);AT 13,20;C$(L+2):
RETURN
640 LET M=M-1: IF M<1 THEN
LET M=M+12
650 PRINT AT 7,10;A$(M);AT 10,
10;A$(M+1);AT 13,10;A$(M+2):
RETURN
660 LET K=K-1: IF K<1 THEN
LET K=K+12
670 PRINT AT 7,15;B$(K);AT 10,
15;B$(K+1);AT 13,15;B$(K+2):
RETURN
```

```
CE PERDEU TODO SEU DINHEIRO":
SOUND 1,-20
800 PRINT '"      QUER RECOMECA
R  (S/N) ?"
810 IF INKEY$="" THEN  GOTO
 810
820 LET I$=INKEY$: IF I$="S"
OR I$="s" THEN  RUN
830 STOP
840 CLS : PRINT AT 10,0;"
      PARABENS !              VO
CE ACABA DE GANHAR O PREMIO.":
 PRINT '" VOCE ESTA $500,00 MA
IS RICO !": FOR J=1 TO 3: FOR
I=1 TO 10: SOUND .01,5*I: NEXT
 I: NEXT J
850 GOTO 800
```

A rotina da linha 760 oferece ao usuário a oportunidade de disputar mais um jogo. Ela é chamada quando o jogador fica sem dinheiro.

A rotina que cuida do prêmio máximo — três sinos — dá as boas novas ao jogador e soma $500 ao seu score. Nesse caso, a partida termina, pois a banca foi quebrada. O jogador pode, então, jogar outra vez.

```
GOTO 400
470 IF I$="S" AND NUDGE=1 THEN
 GOSUB 660: LET NUDGE=0: PRINT
AT 13,26; INK 2;"      ": SOUND
.1,30: GOSUB 510: LET RN=INT (
RND*10): IF INT (RN/2)=RN/2
THEN  LET NUDGE=1: PRINT AT 13
,26; INK 7; BRIGHT 1;"NUDGE":
GOTO 400
480 IF I$="A" AND NUDGE=1 THEN
 GOSUB 640: LET NUDGE=0: PRINT
AT 13,26; INK 2;"      ": SOUND
.1,30: GOSUB 510: LET RN=INT (
RND*10): IF INT (RN/2)<>RN/2
AND RN>6 THEN  LET NUDGE=1:
PRINT AT 13,26; INK 7; BRIGHT
1;"NUDGE": GOTO 400
490 IF HFLAG=1 AND I$="1" OR I
$="2" OR I$="3" OR I$="4" OR I
$="5" OR I$="6" THEN  LET HOLD
=VAL I$: FOR I=1 TO 19 STEP 5:
 PRINT AT 16,I; INK 2;"      ":
 NEXT I: GOTO 220
500 GOTO 400
560 LET M=M+1: IF M>12 THEN
LET M=M-12
570 PRINT AT 7,10;A$(M);AT 10,
10;A$(M+1);AT 13,10;A$(M+2):
RETURN
580 LET K=K+1: IF K>12 THEN
LET K=K-12
590 PRINT AT 7,15;B$(K);AT 10,
15;B$(K+1);AT 13,15;B$(K+2):
RETURN
600 LET L=L+1: IF L>12 THEN
LET L=L-12
```



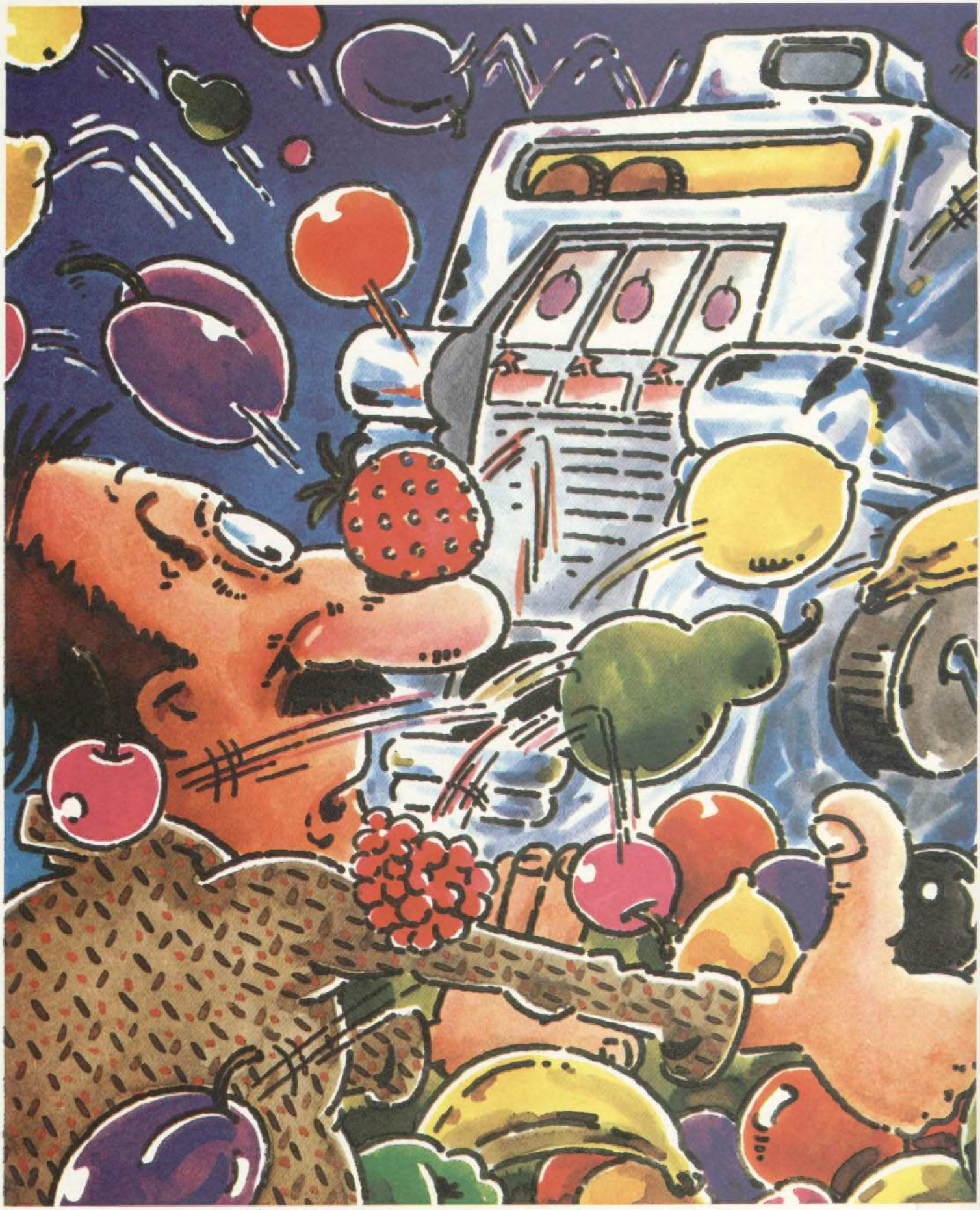
A rotina que empurra as rodas é muito semelhante àquela que as segura. As rodas são movimentadas para cima ou para baixo conforme a escolha do jogador. As teclas que empurram as rodas são mostradas na tela. A cada empurrão, um número aleatório é usado para determinar se o jogador poderá recorrer novamente a essa função.

RAPA-TUDO

```
760 CLS : PRINT AT 10,0;"
     FIM DE JOGO              VO
```

A ROTINA PRINCIPAL

```
350 M=100:H=-1:I=-1:J=-1:P=RND(
16)-1:Q=RND(16)-1:R=RND(16)-1
360 SCREEN 1:GOSUB 1000:GOSUB 2
000:GOSUB 2500:IF M>0 THEN 360
370 CLS:PRINT @101:"YOU RAN OUT
 OF MONEY"
380 PRINT @417," <SPACE> PARA R
ECOMECAR"
390 IF INKEY$<>" " THEN 390 ELS
E RUN
```



A linha 350 estabelece o valor do primeiro cacife em dólar. Ela também acerta os valores iniciais das variáveis que indicam se alguma roda está presa, assim como das variáveis que controlam a posição das rodas. A linha 360, laço principal do programa, chama ordenadamente as sub-rotinas que giram as rodas, desenham as frutas na tela e permitem ao jogador apostar, segurar e empurrar as rodas.

Se o jogador ficar sem dinheiro, a linha 370 terminará o jogo e oferecerá ao usuário a oportunidade de disputar outra partida.

FRUTAS NA TELA

```
500 ON CH+1 GOTO 540,530,560,55
0,570,510,520
510 PUT(XX,YY)-(XX+31,YY+15),B,
PSET:RETURN
520 PUT(XX,YY)-(XX+31,YY+15),C,
PSET:RETURN
530 PUT(XX,YY)-(XX+31,YY+15),A,
PSET:RETURN
540 PUT(XX,YY)-(XX+31,YY+15),BR
,PSET:RETURN
550 PUT(XX,YY)-(XX+31,YY+15),S,
PSET:RETURN
560 PUT(XX,YY)-(XX+31,YY+15),PL
,PSET:RETURN
570 PUT(XX,YY)-(XX+31,YY+15),P,
PSET:RETURN
1000 M=M-10:FOR L=1 TO RND(3)+R
ND(3)
1010 IF H GOSUB 1520
1020 IF I GOSUB 1530
1030 IF J GOSUB 1540
1040 NEXT:IF H THEN SOUND 100,1
1050 FOR L=1 TO RND(3)+RND(3)
1060 IF I GOSUB 1530
1070 IF J GOSUB 1540
1080 NEXT:IF I THEN SOUND 120,1
1090 FOR L=1 TO RND(3)+RND(3)
1100 IF J GOSUB 1540
1110 NEXT:IF J THEN SOUND 140,1
1120 H=-1:I=-1:J=-1:RETURN
1500 CLS:IF D=0 THEN RETURN ELS
E PRINT @166,USING"CREDITO= $$#
#.##";M/100:FOR A=10 TO D STEP
 10:M=M+10:PRINT @166,USING"CRE
DITO= $$###.##";M/100
1510 SOUND 200,1:FOR B=0 TO 400
:NEXT B,A:RETURN
1520 P=(P-1) AND 15:XX=48:YY=28
:FOR G=P-1 TO P+1:CH=R1(15 AND
G):GOSUB 500:YY=YY+32:NEXT:RETU
RN
1530 Q=(Q-1) AND 15:XX=112:YY=2
8:FOR G=Q-1 TO Q+1:CH=R2(15 AND
 G):GOSUB 500:YY=YY+32:NEXT:RET
URN
1540 R=(R-1) AND 15:XX=176:YY=2
8:FOR G=R-1 TO R+1:CH=R3(15 AND
 G):GOSUB 500:YY=YY+32:NEXT:RET
URN
1550 C=9:IF (R1(P)=R2(Q)) AND(R
2(Q)=R3(R)) THEN C=R1(P):RETURN
1560 IF R1(P)=R2(Q) AND (R1(P)=
6 OR R1(P)=5) THEN C=1+R1(P):RE
TURN
1570 IF R1(P)=6 THEN C=8
1580 RETURN
```

As linhas 500 a 570 usam **PUT** para desenhar as frutas na tela. As linhas 1000 a 1120 giram as rodas, chamando as sub-rotinas das linhas 1520 a 1540. As linhas 1150 a 1180 acusam um resultado premiado e a linha 1500 soma o prêmio às posses do jogador.

APOSTAR, SEGURAR, EMPURRAR

```
2000 GOSUB 1550
2010 IF C=9 OR C=0 THEN D=W(C):
GOSUB 1500:RETURN
2020 CLS9-C:PRINT @265,"apostar
";:PRINT@278,USING "$$#.## ";W(
C)/100;
2030 PLAY "L4T20B":PRINT @212,U
SING"$$#.## ";W(C-1)/100;:PRINT
 @340,STRING$(7,271-C*16);
2040 PLAY"T20C":PRINT @212,STRI
NG$(7,271-C*16);:PRINT @340,USI
NG"$$#.## ";W(C+1)/100;
2050 R$=INKEY$:IF R$<>" " AND R
$<>CHR$(13) THEN 2030
2060 IF R$=CHR$(13) THEN CLS:D=
W(C):GOSUB 1500:RETURN
2070 IF RND(2)=1 THEN CLS:D=W(C
+1):GOSUB 1500:RETURN
2080 C=C-1:IF C=0 THEN D=200:GO
SUB 1500:RETURN
2090 GOTO 2020
2500 IF RND(4)=1 GOSUB 3060:GOT
O 2550
2510 IF RND(5)<3 THEN 2560
2520 FOR K=1 TO 2000:NEXT:SCREE
N 1,0
2530 A$=INKEY$:IF A$<>" " AND A
$<>"C" THEN 2530
2540 IF A$=" " THEN RETURN
2550 CLS:PRINT @166,USING"CREDI
TO= $$###.##";M/100:GOTO 2520
2560 SCREEN 1,0:H=-1:I=-1:J=-1
2570 IF H THEN PUT(38,122)-(91,
143),H,NOT
2580 IF I THEN PUT(102,122)-(15
5,143),H,NOT
2590 IF J THEN PUT(166,122)-(21
9,143),H,NOT
2600 R$=INKEY$:IF R$=" " THEN F
OR K=0 TO 2:PUT(38+64*K,122)-(9
1+64*K,143),H,PSET:NEXT:RETURN
3000 IF R$<"1" OR R$>"4" THEN 2
570
3010 ON VAL(R$) GOTO 3020,3030,
3040,3050
3020 H=-1:I=-1:J=-1:GOTO 2570
3030 H=0:PUT(38,122)-(91,143),H
,PSET:GOTO 2570
3040 I=0:PUT(102,122)-(155,143)
,H,PSET:GOTO 2570
3050 J=0:PUT(166,122)-(219,143)
,H,PSET:GOTO 2570
3060 SCREEN 1,0:COLOR 4,2:PUT(1
59,156)-(224,170),H,NOT:PLAY"L4
0T10"
3070 K=1
3080 LINE(10+K*16,158)-(21+K*16
,169),PRESET,BF
3090 IF INKEY$=" " THEN 3120
3100 K=K+1:PLAY STR$(K*2):IF K<
6 THEN 3080
3110 FOR K=1 TO 5:LINE(10+K*16,
158)-(21+K*16,169),PSET,BF:NEXT
:GOTO 3070
3120 N=K:PUT(159,156)-(224,170)
,H,NOT
3130 R$=INKEY$:IF(R$<"5" OR R$>
"9") AND R$<>"0" THEN 3130
3140 IF R$="0" THEN R$="10"
3150 ON VAL(R$)-4 GOTO 3160,317
0,3180,3190,3200,3210
3160 P=P+2:GOSUB 1520:GOTO 3220
3170 Q=Q+2:GOSUB 1530:GOTO 3220
3180 R=R+2:GOSUB 1540:GOTO 3220
3190 GOSUB 1520:GOTO 3220
3200 GOSUB 1530:GOTO 3220
3210 GOSUB 1540
3220 SOUND 40,1:GOSUB 1550:IF C
<9 GOSUB 2010:N=0:GOTO 3250
3230 IF N=1 THEN N=0:GOTO 3250
3240 LINE(10+N*16,158)-(21+N*16
,169),PSET,BF:N=N-1:GOTO 3130
3250 FOR K=1 TO 5:LINE(10+K*16,
158)-(21+K*16,169),PSET,BF:NEXT
:RETURN
```

As linhas 2010 a 2050 permitem que o jogador faça a sua aposta. Ele pode então melhorar o prêmio que recebeu anteriormente (ou, na pior das hipóteses, perder parte dele).

A rotina que segura as rodas encontra-se entre as linhas 2530 e 3050, sendo chamada da linha 2510. Os indicadores que definem a liberdade das rodas são modificados de acordo com as teclas apertadas pelo jogador.

As linhas 3060 a 3250 permitem que o jogador empurre uma das rodas, movendo-a uma posição para cima ou para baixo. O número de movimentos oferecidos é selecionado ao acaso na linha 2500. A rotina detecta as teclas selecionadas, movendo as rodas de acordo com cada uma delas.

A ROTINA PRINCIPAL

```
1160 FOR I=1 TO 2000:NEXT
1200 R=RND(-TIME):M=100:H=-1:I=
-1:J=-1:P=INT(RND(1)*15):Q=INT(
RND(1)*15):R=INT(RND(1)*15)
1210 LOCATE 0,22:PRINT STRING$(
30,32);:GOSUB 1400:GOSUB 2000:G
OSUB 2500:IF M>0 AND M<5000 THE
N 1210
1215 IF M>=5000 THEN CLS:LOCATE
 0,11:PRINT "RAPA TUDO E ESTOUR
A A BANCA":GOTO 1230
1220 LOCATE 0,22:PRINT "SEU DIN
HEIRO ACABOU          ";
1230 LOCATE 0,23:PRINT "APERTE
>ESPACO P/ REPETIR";
1240 IF INKEY$<>" " THEN 1240 E
LSE RUN
```

A linha 1200 estabelece o valor de um dólar para o primeiro cacife e acerta os valores iniciais das variáveis que indicam se alguma das rodas está presa, assim como das variáveis que controlam a posição das rodas. A linha 1210 é o laço principal do programa. Ela chama ordenadamente as sub-rotinas que giram as rodas, desenham as frutas na tela e permitem ao jogador apostar, segurar e empurrar as rodas. O gerador de números randômicos também é acertado nessa linha.

Se o jogador ficar sem dinheiro, a linha 1220 concluirá o jogo e dará ao usuário a oportunidade de jogar novamente. Quando existirem três sinos na linha correspondente ao meio das rodas, haverá o estorno da banca e o jogo também terá chegado ao fim, agora com a vitória do jogador.

FRUTAS NA TELA

```
1300 VPOKE BASE(5)+YY,C1:VPOKE
BASE(5)+YY+1,C2:RETURN
1400 M=M-10:GOSUB 4000:FOR L=1
TO INT(RND(1)*3+1)*INT(RND(1)*3
+1)
1405 IF H THEN GOSUB 1520
1410 IF I THEN GOSUB 1530
1415 IF J THEN GOSUB 1540
1420 NEXT
1425 FOR L=1 TO INT(RND(1)*3+1)
*INT(RND(1)*3+1)
1430 IF I THEN GOSUB 1530
1435 IF J THEN GOSUB 1540
1440 NEXT
1450 FOR L=1 TO INT(RND(1)*3+1)
*INT(RND(1)*3+1)
1455 IF J THEN GOSUB 1540
1460 NEXT
1470 FOR L=8 TO 10:SOUND L,0:NE
XT:H=-1:I=-1:J=-1
1480 RETURN
1500 IF D=0 THEN RETURN ELSE M=
M+D
1510 RETURN
1520 P=(P-1)AND15:YY=235:FOR G=
P-1 TO P+1:C1=A(15ANDG,1):C2=A(
15ANDG,2):GOSUB 1300:YY=YY+96:N
EXT:RETURN
1530 Q=(Q-1)AND15:YY=239:FOR G=
Q-1 TO Q+1:C1=B(15ANDG,1):C2=B(
15ANDG,2):GOSUB 1300:YY=YY+96:N
EXT:RETURN
1540 R=(R-1)AND15:YY=243:FOR G=
R-1 TO R+1:C1=C(15ANDG,1):C2=C(
15ANDG,2):GOSUB 1300:YY=YY+96:N
EXT:RETURN
1550 D=0
1551 IF A(P,1)=136 THEN D=10
1552 IF A(P,1)=B(Q,1) AND A(P,1
)=136 THEN D=20:RETURN
1553 IF NOT(A(P,1)=B(Q,1) AND B
(Q,1)=C(R,1)) THEN RETURN
1554 A=A(P,1)
1556 IF A=120 THEN D=50
1557 IF A=184 THEN D=50
1558 IF A=216 THEN D=50
1559 IF A=152 THEN D=100
1560 IF A=168 THEN D=100
1561 IF A=200 THEN M=M+5000:GOT
O 1215
1570 RETURN
```

A linha 1300 usa o comando **VPOKE** para desenhar as frutas na tela. As linhas que vão de 1400 a 1480 giram as rodas chamando as sub-rotinas das linhas 1520 a 1540.

As linhas 1550 a 1570, por sua vez, verificam se houve um resultado premiado e a linha 1500, finalmente, soma o prêmio às posses do jogador.

EMPURRAR E SEGURAR AS RODAS

```
2000 GOSUB 1550:GOSUB 1500:GOSU
B 2550:RETURN
2500 IF INT(RND(1)*4+1)=1 THEN
GOSUB 3060:GOSUB 2550:GOTO 2530
2510 IF INT(RND(1)*5+1)<3 THEN
2560
2530 A$=INKEY$:IF A$<>" " THEN
2530
2540 RETURN
2550 M$=STR$(M):LOCATE 2,15:PRI
NT RIGHT$(M$,2);:LOCATE 2,14:PR
INT "  ";:IF M>=100THEN LOCATE
2,14:PRINT LEFT$(M$,2);:RETURN
2555 RETURN
2560 H=-1:I=-1:J=-1:LOCATE 4,22
:PRINT "PODE SEGURAR AS RODAS"
2600 R$=INKEY$:IF (R$<"1" OR R$
>"6") AND R$<>" " THEN 2560
3000 IF R$=" " THEN RETURN
3010 ON VAL(R$) GOTO 3020,3025,
3030,3035,3040,3045
3020 H=0:GOTO 3050
3025 I=0:GOTO 3050
3030 J=0:GOTO 3050
3035 H=0:I=0:GOTO 3050
3040 J=0:I=0:GOTO 3050
3045 J=0:H=0
3050 RETURN
3060 LOCATE 3,22:PRINT "PODE EM
PURRAR AS RODAS"
3070 N=INT(RND(1)*5+1)
3080 R$=INKEY$:IF R$="" THEN 30
80
3100 IF R$="Q" THEN 3160
3110 IF R$="W" THEN 3170
3120 IF R$="E" THEN 3180
3130 IF R$="A" THEN 3190
3140 IF R$="S" THEN 3200
3150 IF R$="D" THEN 3210
3155 GOTO 3080
3160 P=P+2:GOSUB 1520:GOTO 3220
3170 Q=Q+2:GOSUB 1530:GOTO 3220
3180 R=R+2:GOSUB 1540:GOTO 3220
3190 GOSUB 1520:GOTO 3220
3200 GOSUB 1530:GOTO 3220
3210 GOSUB 1540
3220 GOSUB 1550:IF D>0 THEN GOS
UB 2000:N=0:GOTO 3250
3230 IF N=1 THEN N=0:GOTO 3250
3240 N=N-1:GOTO 3080
3250 LOCATE 0,22:PRINT STRING$(
30,32);:RETURN
```

A rotina que segura as rodas está entre as linhas 2560 e 3050, sendo chamada a partir da linha 2510. Os indicadores que revelam se as rodas estão livres são modificados conforme as teclas apertadas pelo jogador.

As linhas que vão de 3060 a 3250 permitem que o jogador empurre uma das rodas, movendo-a uma posição para cima ou para baixo. O número de empurrões oferecidos é selecionado ao acaso na linha 3070. A rotina detecta as teclas selecionadas, movendo as rodas de acordo com cada uma delas.

EFEITOS SONOROS

A rotina situada na linha 4000 acerta os valores dos registros do chip sonoro do MSX para simular o ruído de rodas em movimento.

```
4000 RESTORE 4030:FOR L=4 TO 13
4010 READ A:SOUND L,A
4020 NEXT
4030 DATA 0,9,15
4040 DATA 42,0,12,16,100,5,12
4050 RETURN
```

```
300  GOSUB 700: GOSUB 4000: GOS
UB 310: GOSUB 780: GOTO 1000
1000 M = 1:H = 1:I = 1:J = 1:P
=  INT ( RND (1) * 16):Q =  INT
 ( RND (1) * 16):R =  INT ( RND
 (1) * 16)
1010  POKE  - 16299,0: POKE  -
16304,0: GOSUB 1300: GOSUB 2000
: GOSUB 2500: IF M > 0 THEN 101
0
1020  TEXT : HOME : VTAB 20: PR
INT "SEU DINHEIRO ACABOU"
1030  PRINT "APERTE A BARRA DE
ESPACO": PRINT "PARA JOGAR NOVA
MENTE"
1040  GET A$: IF A$ = " " THEN
 RUN
1050  GOTO 1040
1200  HCOLOR= 0: DRAW 15 AT XX,
YY: DRAW 15 AT XX + 8,YY: DRAW
15 AT XX,YY + 4: DRAW 15 AT XX
+ 8,YY + 4
1205  ON CH + 1 GOTO 1240,1230,
1260,1250,1270,1210,1220
1210  HCOLOR= 3: DRAW 7 AT XX,Y
Y: DRAW 8 AT XX + 7,YY: RETURN
1220  HCOLOR= 3: DRAW 5 AT XX,Y
Y: DRAW 6 AT XX + 7,YY: RETURN
1230  HCOLOR= 7: DRAW 13 AT XX,
YY: DRAW 14 AT XX + 7,YY: RETUR
N
1240  HCOLOR= 3: DRAW 3 AT XX,Y
Y: DRAW 4 AT XX + 8,YY: RETURN
1250  HCOLOR= 3: DRAW 1 AT XX +
 1,YY: DRAW 2 AT XX + 8,YY: RET
URN
1260  HCOLOR= 7: DRAW 11 AT XX,
YY: DRAW 12 AT XX + 7,YY: RETUR
N
```

```
1270  HCOLOR= 3: DRAW 9 AT XX +
 1,YY: DRAW 10 AT XX + 8,YY: RE
TURN
1300 M = M - .1: FOR L = 1 TO
INT (3 *  RND (1) + 1) +  INT (
3 *  RND (1) + 1)
1310  IF H THEN  GOSUB 1520:X =
  PEEK ( - 16336)
1320  IF I THEN  GOSUB 1530:X =
  PEEK ( - 16336)
1330  IF J THEN  GOSUB 1540:X =
  PEEK ( - 16336)
1340  NEXT : IF H THEN  CALL  -
 198
1350  FOR L = 1 TO  INT (3 *  R
ND (1) + 1) +  INT (3 *  RND (1
) + 1)
1360  IF I THEN  GOSUB 1530:X =
  PEEK ( - 16336)
1370  IF J THEN  GOSUB 1540:X =
  PEEK ( - 16336)
1380  NEXT : IF I THEN  CALL  -
 198
1390  FOR L = 1 TO  INT (3 *  R
ND (1) + 1) +  INT (3 *  RND (1
) + 1)
1400  IF J THEN  GOSUB 1540:X =
  PEEK ( - 16336)
1410  NEXT : IF J THEN  CALL  -
 198
1420 H = 1:I = 1:J = 1: RETURN
1500  IF D = 0 THEN  RETURN
1505  HOME : TEXT : VTAB 20:M =
 M + D: PRINT "CREDITO: $";M
1510  FOR B = 0 TO 400: NEXT :
RETURN
1520 P = P - 1:XX = 80:YY = 90:
 IF P =  - 1 THEN P = 15
1522  FOR G = P - 1 TO P + 1: I
F G =  - 1 THEN CH = R1(15): GO
TO 1525
1523  IF G = 16 THEN CH = R1(0)
: GOTO 1525
1524 CH = R1(G)
1525  GOSUB 1200:YY = YY + 20:
NEXT : RETURN
1530 Q = Q - 1:XX = 128:YY = 90
: IF Q =  - 1 THEN Q = 15
1532  FOR G = Q - 1 TO Q + 1: I
F G =  - 1 THEN CH = R2(15): GO
TO 1535
1533  IF G = 16 THEN CH = R2(0)
: GOTO 1535
1534 CH = R2(G)
1535  GOSUB 1200:YY = YY + 20:
NEXT : RETURN
1540 R = R - 1:XX = 174:YY = 90
: IF R =  - 1 THEN R = 15
1542  FOR G = R - 1 TO R + 1: I
F G =  - 1 THEN CH = R3(15): GO
TO 1545
1543  IF G = 16 THEN CH = R3(0)
: GOTO 1545
1544 CH = R3(G)
1545  GOSUB 1200:YY = YY + 20:
NEXT : RETURN
1550 C = 9: IF (R1(P) = R2(Q))
AND (R2(Q) = R3(R)) THEN C = R1
(P): RETURN
1560  IF R1(P) = R2(Q) AND (R1(
P) = 6 OR R1(P) = 5) THEN C = 1
 + R1(P): RETURN
1570  IF R1(P) = 6 THEN C = 8
1580  RETURN
2000  GOSUB 1550
2010  IF C = 9 OR C = 0 THEN D
= W(C): GOSUB 1500: RETURN
2020  HOME : TEXT : VTAB 20: PR
INT "PREMIO = $";W(C)
2030  PRINT "QUER APOSTAR ?
2050  PRINT "<ESPACO>"; TAB( 20
);"APOSTA": PRINT "<RETURN>"; T
AB( 20);"RECOLHE O PREMIO"
2055  GET A$: IF A$ <  > " " AN
D A$ <  >  CHR$ (13) THEN 2020
2060  IF A$ =  CHR$ (13) THEN
HOME :D = W(C): GOSUB 1500: RET
URN
2070  IF  INT (2 *  RND (1) + 1
) = 2 THEN  HOME :D = W(C + 1):
 GOSUB 1500: RETURN
2080 C = C - 1: IF C = 0 THEN D
 = 2: GOSUB 1500: RETURN
2090  GOTO 2020
2500  IF  INT (4 *  RND (1) + 1
) = 1 THEN  GOSUB 3060: GOTO 25
50
2510  IF  INT (5 *  RND (1) + 1
) < 3 THEN 2560
2530  GET A$: IF A$ <  > "C" TH
EN  RETURN
2550  HOME : TEXT : VTAB 20: PR
INT "CREDITO  = $";M: GOTO 2530
2560 H = 1:I = 1:J = 1
2565  HOME : TEXT
2570  IF  NOT H THEN  VTAB 20:
HTAB 10: PRINT "HOLD";
2580  IF  NOT I THEN  VTAB 20:
HTAB 20: PRINT "HOLD";
2590  IF  NOT J THEN  VTAB 20:
HTAB 30: PRINT "HOLD";
2595  VTAB 22: HTAB 1: PRINT "V
OCE PODE PRENDER AS RODAS";
2597  FOR K = 1 TO 4000: NEXT :
 POKE  - 16299,0: POKE  - 16304
,0: FOR K = 1 TO 4000: NEXT : P
OKE  - 16300,0: POKE  - 16303,0
2600  GET A$: IF A$ = " " THEN
 VTAB 20: PRINT " ": RETURN
3000  IF A$ < "1" OR A$ > "4" T
HEN 2570
3010  ON  VAL (A$) GOTO 3020,30
30,3040,3050
3020 H = 1:I = 1:J = 1: GOTO 25 65
3030 H = 0: GOTO 2565
3040 I = 0: GOTO 2565
3050 J = 0: GOTO 2565
3060  HOME : TEXT : VTAB 22: PR
INT "PODE EMPURRAR AS RODAS";:
GET A$
3070 N =  INT (5 *  RND (1) + 1
)
3080  POKE  - 16299,0: POKE  -
16304,0: GET A$: IF (A$ < "5" O
R A$ > "9") AND A$ <  > "0" THE
N 3060
3140  IF A$ = "0" THEN A$ = "10
"
3150  ON  VAL (A$) - 4 GOTO 316
0,3170,3180,3190,3200,3210
3160 P = P + 2: GOSUB 1520:X =
 PEEK ( - 16336): GOTO 3220
3170 Q = Q + 2: GOSUB 1530:X =
 PEEK ( - 16336): GOTO 3220
3180 R = R + 2: GOSUB 1540:X =
 PEEK ( - 16336): GOTO 3220
3190  GOSUB 1520:X =  PEEK ( -
16336): GOTO 3220
3200  GOSUB 1530:X =  PEEK ( -
16336): GOTO 3220
3210  GOSUB 1540
3220  GOSUB 1550: IF C < 9 THEN
  GOSUB 2010:N = 0: GOTO 3250
3230  IF N = 1 THEN N = 0: GOTO
 3080
3240 N = N - 1: GOTO 3080
3250  RETURN
```

A linha 1000 estabelece que o valor do cacife inicial é um dólar. Ela também acerta os valores iniciais das variáveis que indicam se alguma roda está presa, assim como das variáveis que controlam a posição das rodas. A linha 1010, laço principal do programa, chama ordenadamente as sub-rotinas que giram as rodas, desenham as frutas na tela e permitem ao jogador apostar, segurar e empurrar as rodas.

Se o jogador ficar sem dinheiro, a linha 1020 terminará o jogo, permitindo-lhe jogar novamente.

As linhas 1200 a 1270 usam **DRAW** para desenhar as frutas na tela. As linhas 1300 a 1420 giram as rodas, chamando as sub-rotinas das linhas 1520 a 1540. As linhas 1550 a 1580 acusam um resultado premiado e a 1500 soma o prêmio às posses do jogador.

As linhas 2010 a 2050 permitem que o jogador aposte. Este pode então melhorar o prêmio que recebeu (ou perder parte dele). A rotina que segura as rodas se encontra entre as linhas 2530 e 3050, sendo chamada a partir da linha 2510. Os indicadores que revelam se as rodas podem girar livremente são modificados conforme as teclas apertadas pelo jogador.

As linhas 3060 a 3250 possibilitam ao jogador empurrar uma das rodas, movendo-a uma posição para cima ou para baixo. O número de empurrões oferecidos é selecionado ao acaso na linha 2500. A rotina detecta as teclas selecionadas, movendo as rodas de acordo com cada uma destas.

Para funcionar no TK-2000, o programa precisa de algumas modificações.

Elimine todos os **POKE −16299,0** e **POKE −16304,0**. Substitua os comandos **HOME:TEXT** — sempre que aparecerem assim juntos — por **GOSUB 5000**.

Acrescente ainda as linhas:

```
5000  FOR K = 160 TO 191
5010  HCOLOR= 0
5020  HPLOT 0,K TO 279,K
5030  NEXT : RETURN
```

MENSAGENS SECRETAS

6 de junho de 1944: os acordes iniciais da Quinta Sinfonia de Beethoven dão o sinal para a invasão da Normandia pelos aliados, num dos casos mais famosos de mensagem em código.

Embora sem saber, a maioria das pessoas usa códigos para comunicar-se em seu dia-a-dia. Assim, quando alguém pede um sapato número 39 ou fala de um microprocessador Z80, está empregando um código. Do mesmo modo, um cientista nuclear usa equações muito complicadas, que, na verdade, não passam de códigos, para simplificar a representação de fenômenos complexos. Nós poderíamos obter os mesmos resultados simplesmente descrevendo tais fenômenos por meio de palavras. No entanto, isso nos tomaria muito mais tempo.

Nos exemplos citados, a preocupação maior consiste em facilitar a comunicação e não em esconder a informação transmitida. Os códigos podem também ser usados para economizar gastos ou espaços de armazenamento. Imaginemos, por exemplo, uma companhia que realiza muitos contratos envolvendo cláusulas e parágrafos iguais. Se todos esses detalhes fossem guardados em fitas ou discos, economizaríamos certamente muito espaço, codificando as partes que se repetem várias vezes.

A ciência que estuda as mensagens secretas chama-se criptografia. Seu nome é formado a partir de duas palavras gregas — *Kryptos* (secreto) e *graphos* (escrita) —, e refere-se tanto à escrita codificada quanto à cifrada. Os termos código e cifra têm, na realidade, significado ligeiramente diferente no que diz respeito ao modo como a mensagem é transmitida. Quando a informação é comunicada letra por letra, dizemos que ela está cifrada. Quando uma ou mais palavras são transformadas em outro grupo de expressões ou de números, por intermédio de alguma espécie de dicionário, dizemos que ela está codificada. Na prática, o termo codificar é usado em ambos os casos.

CÓDIGOS SECRETOS

A codificação e decodificação de mensagens secretas já foram de uso restrito dos militares, ou pelo menos dos Serviços de Inteligência. Hoje, porém, o uso de linhas públicas de telefone e de satélites para transmissão de informações confidenciais fez aumentar enormemente a necessidade dos códigos.

Uma das primeiras aplicações importantes dos computadores aconteceu durante a Segunda Guerra Mundial, quando equipamentos IBM foram utilizados pelos Aliados para decifrar mensagens em código do inimigo. Desde então, espiões e contra-espiões das grandes potências têm acompanhado atentamente os avanços tecnológicos nesse campo.

Atualmente, as possibilidades abertas pela computação são tantas que jovens pouco escrupulosos — denominados *hackers* — ligam seus micros a redes telefônicas e penetram facilmente em sistemas bancários e instalações militares, quebrando códigos e alterando informações.

Neste e no próximo artigo mostraremos como empregar o computador para criar mensagens em código secreto, recorrendo a diversos métodos que, como cada tipo de espionagem, têm diferentes níveis de sofisticação.

Mesmo não sendo um agente internacional, você pode lançar mão desses métodos quando quiser enviar mensagens confidenciais a amigos que também tenham micros. E aqui vai um desafio. Neste artigo está escondida uma informação codificada. Tente encontrá-la.

CÓDIGOS DE POSIÇÃO

Os símbolos mostrados nas telas de vídeo da página 889 parecem não ter significado algum, mas na realidade são exemplos de códigos de posição. Como o nome sugere, esse é um código baseado na posição de um símbolo em relação a um ponto dado.

Esse tipo de código foi usado pela primeira vez há mais de 2000 anos por Lisandro, um almirante espartano que, durante uma das batalhas da Guerra do Peloponeso (travada entre as cidades gregas Esparta e Atenas), percebeu que, no cinto de um de seus escravos, a distância dos furos até a fivela continha na realidade uma mensagem secreta. Desde então, os códigos tornaram-se um dos recursos mais poderosos da chamada "arte de guerra".

Você pode usar um tipo de código de posição simplesmente escrevendo o alfabeto no alto de uma folha e colocando abaixo dele as distâncias recebidas, conforme mostram as figuras 1 e 2. Como você tem a letra-chave na primeira linha, é fácil decifrar a mensagem. Para dificultar sua decodificação, basta fazer uma rotação nas letras-chave (veja a figura 3). Você pode, por exemplo, começar com N e, quando chegar no Z, começar de novo com A. Isto é chamado rotação cíclica.

O primeiro programa usa esse método para produzir uma versão codificada de seu texto — entretanto, não devem ser deixados espaços em branco entre as palavras. Se você quiser realmente enviar sua mensagem para alguém, vai precisar de uma impressora.

- CÓDIGOS E CIFRAS
- A CRIPTOGRAFIA
- CÓDIGOS SECRETOS E SUAS APLICAÇÕES
- COMO PRODUZIR SEUS PRÓPRIOS CÓDIGOS SECRETOS
- CÓDIGOS DE POSIÇÃO
- CÓDIGO DE JÚLIO CÉSAR
- CIFRA SAINT CYR
- CÓDIGO MORSE

```
20 BORDER 0: PAPER 0: INK 7:
CLS
30 PRINT TAB 6;" Codigo de po
sicao ": PRINT
40 PRINT  INK 2; PAPER 7;
FLASH 1;AT 6,10;"  CUIDADO  "
: PRINT
50 PRINT "Nao deixe espacos e
ntre palavras"
60 PRINT : PRINT "       Escre
va em minusculas"
70 INPUT "Qual a mensagem ?"'
a$
80 FOR i=1 TO 400: NEXT i:
CLS
90 FOR i=1 TO LEN (a$)
100 LET b$=a$(i)
110 LET v=CODE (b$)-96
120 IF v<=32 THEN  PRINT TAB v
; INK 6;"*": GOTO 150
130 LET v=v-26
140 PRINT TAB (v); INK 6;"*"
150 NEXT i
```

```
20 CLS
30 PRINT @6,"CODIGO DE DISTANCI
A"
40 PRINT @140,"CUIDADO":PRINT
50 PRINT"NAO DEIXE ESPACOS ENTR
E PALAVRAS"
60 PRINT:PRINT
70 PRINT"QUAL A MENSAGEM ?":INP
UT A$
80 FOR I=1 TO 600:NEXT:CLS
90 FOR I=1 TO LEN(A$)
```

1

Código de posição com distância horizontal

2

Código de posição com distância vertical

```
100 B$=MID$(A$,I,1)
110 V=ASC(B$)-45
120 IF V<=32 THEN PRINT TAB(V)
"*":GOTO 150
130 V=V-26
140 PRINT TAB(V)"*"
150 NEXT
```

```
20 CLS
30 PRINT TAB(10) "código de pos
ição":PRINT
40 PRINT TAB(14) "cuidado":PRIN
T
50 PRINT TAB(4) "não deixe espa
ço entre as palavras"
60 PRINT:PRINT
70 PRINT"qual a sua mensagem?":
INPUT A$
80 FOR I=1 TO 600:NEXT I:CLS
90 FOR I=1 TO LEN (A$)
100 B$=MID$(A$,I,1)
110 V=ASC(B$)-45
120 IF V<=32 THEN PRINT TAB(V)"
*":GOTO 30
130 V=V-26
```

```
140 PRINT TAB(V)"*"
150 NEXT I
```

```
5  HOME
10  PRINT  TAB( 9)"CODIGO DE DI
STANCIA": PRINT
20  PRINT  TAB( 14)"CUIDADO": P
RINT
30  PRINT  TAB( 4)"NAO DEIXE ES
PACO  ENTRE AS PALAVRAS"
40  PRINT : PRINT
50  PRINT "QUAL A SUA  MENSAGEM
 ": INPUT A$
60  FOR I = 1 TO 600: NEXT I: H
OME
70  FOR I = 1 TO  LEN (A$)
80 B$ =  MID$ (A$,I,1)
90 V =  ASC (B$) - 45
100  IF V <  = 32 THEN  PRINT
TAB( V)"*": GOTO 130
110 V = V - 26
120  PRINT TAB( V)"*"
130  NEXT I
```

Inicialmente, o programa descobre o valor ASCII de cada letra de seu texto, através do comando **MID$** no laço que vai da linha 90 à linha 150 (130 no Apple e no TK-2000). Uma vez que a mensagem já foi convertida em uma série de números, é fácil fazer a codificação por meio de uma transformação linear. No caso do MSX, por exemplo, a letra que está na variável **V** será transformada na distância que é o seu valor em ASCII menos 26 (linha 130). O comando **TAB** imprimirá o asterisco na distância desejada, a partir do lado direito da tela (linha 120), e o processo de codificação estará pronto.

COMO USAR O CÓDIGO

Apesar de parecer demasiado simples para ser eficiente, o código de posição conta com vários fatores a seu favor. Em primeiro lugar, antes de decifrar um código, você deve verificar se ele realmente existe. Isso porque, sendo constituída de vários pontos (ou asteriscos) dispostos aparentemente de forma aleatória sobre qualquer superfície, uma mensagem assim codificada facilmente passaria despercebida. Durante a Segunda Guerra, aliás, esse truque foi usado por agentes de vários países. Assim, após uma inspeção mais cuidadosa na inocente fotografia de um quintal, descobriu-se que as estacas de um varal dispunham-se de maneira a formar uma mensagem secreta.

Um modo de fazer com que o código de posição fique mais difícil de ser quebrado é reestruturar o programa, tornando as letras-chave realmente aleatórias. Sem essa alteração, um perito

3

Cifra de Saint Cyr

que saiba estar manipulando um código de posição terá de tentar no máximo 26 combinações antes de resolver o problema. Se a ordem das letras for aleatória, o número de combinações crescerá extraordinariamente.

CIFRA DE SAINT CYR

Na Antiguidade, os maiores criptógrafos depois dos gregos foram os romanos. Júlio César, por exemplo, inventou um código no qual cada letra era substituída pela terceira letra que se lhe seguia no alfabeto. Nesse caso, o A torna-se D, o B torna-se E, e assim por diante. No fim do alfabeto, o X torna-se A, o Y, B e o Z é substituído pelo C. Usando esse método, a mensagem MINAR CAMPO LADO LESTE cifrada seria: PLQDU FDPSR ODGR OHVWH.

Como você verá mais tarde, o código criado por Júlio César é um caso especial da chamada cifra de Saint Cyr e você poderá verificar aquela mensagem rodando o próximo programa e usando 3333333 como número-chave.

Com o fim do Império Romano, a criptografia passou por longo período de estagnação e, apesar do crescente uso de mensagens cifradas nos séculos XVI e XVII, só no século XIX a Academia Militar Francesa de Saint Cyr produziu inovações sobre o código de César. A cifra de Saint Cyr é constituída de uma seqüência de letras em ordem alfabética, abaixo da qual colocamos um alfabeto, a partir de um ponto desejado. Nesse caso, cada letra do alfabeto de baixo terá sua correspondente na linha de cima. Assim, como mostra a ilustração desta página, a palavra INPUT (na linha de baixo) seria cifrada como AFHML. Uma das vantagens da cifra de Saint Cyr é que podemos escolher um ponto de partida diferente para cifrar cada letra, bastando que o receptor conheça esses pontos. Isto o torna ainda mais difícil de ser decifrado.

No programa *Cifra de Saint Cyr* incorporou-se a esse código um número-chave, a fim de conferir-lhe maior segurança. Mesmo que consiga ter acesso à listagem do programa, nenhum "espião" decifrará a mensagem, a menos que ele conheça o número secreto.

```
 20 BORDER 0: PAPER 0: INK 7:
 CLS
 25 POKE 23658,8
 30 PRINT TAB 8;"CIFRA DE ST.C
 YR": PRINT
 40 PRINT  INK 2; PAPER 7;
 FLASH 1;AT 6,10;"  CUIDADO  "
 50 PRINT : PRINT "Nao deixe e
 spacos entre palavras"
 60 PRINT : PRINT
 70 PRINT "1"  1 -> codificar"
 80 PRINT " -1 -> decodificar"
 90 INPUT s
100 INPUT "Qual a mensagem ?"'
a$
110 PAUSE 50: CLS
120 INPUT "Qual o numero-chave
? (7 digitos)"'n$
130 PAUSE 50: CLS
140 FOR k=1 TO LEN a$
150 LET l=k-INT (k/7)*7+1
160 LET t=CODE (a$(k))+(s*VAL
(n$(l)))
170 IF t>90 OR t<65 THEN  LET
t=t-(s*26)
180 PRINT CHR$ (t);
190 NEXT k
```

```
20 CLS
30 PRINT @7,"CIFRA DE ST. CYR"
40 PRINT @140,"CUIDADO":PRINT
50 PRINT"NAO DEIXE ESPACOS ENTR
E PALAVRAS"
60 PRINT:PRINT
70 PRINT" < 1> PARA CODIFICAR"
80 PRINT" <-1> PARA DECODIFICAR
"
90 INPUT S
100 INPUT"QUAL A MENSAGEM";A$
110 FOR K=1 TO 1500:NEXT:CLS
120 INPUT"QUAL O NUMERO CHAVE (
7 DIGITOS) ";N$
130 FOR K=1 TO 1500:NEXT:CLS
140 FOR K=1 TO LEN(A$)
150 L=K-INT(K/7)*7+1
160 T=ASC(MID$(A$,K,1))+(S*VAL(
MID$(N$,L,1)))
170 IF T>90 OR T<65 THEN T=T-(S
*26)
180 PRINT CHR$(T);
190 NEXT
```

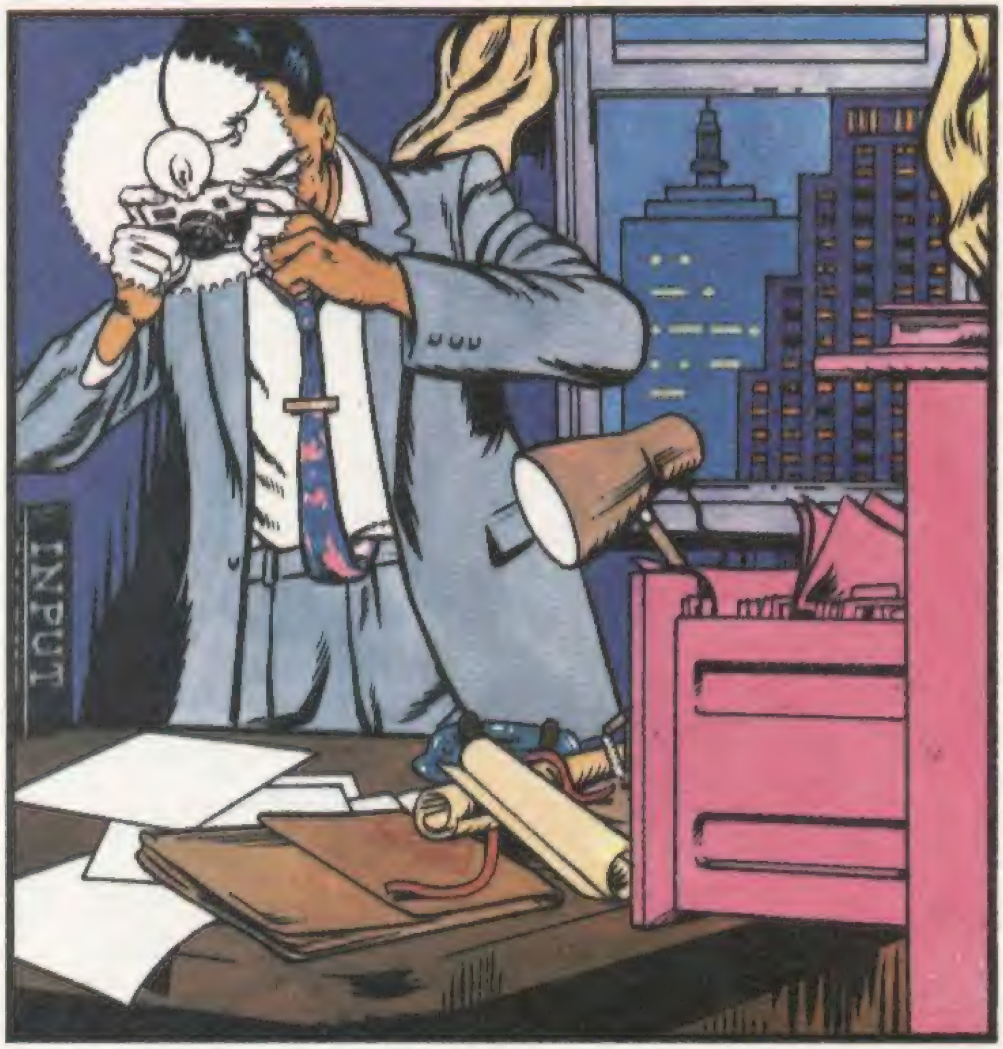
```
20 CLS
30 PRINT TAB(10)"CIFRA SAINT CY
R":PRINT
40 PRINT TAB(14)"CUIDADO":PRINT
50 PRINT TAB(2)"NÃO DEIXE ESPAÇ
 O ENTRE AS PALAVRAS"
```

```
60 PRINT:PRINT
70 PRINT"DIGITE  1 PARA CODIFIC
AR"
80 PRINT"DIGITE -1 PARA DECODIF
ICAR"
90 INPUT S
100 INPUT"QUAL SUA MENSAGEM";A$
110 FOR K=1 TO 1500:NEXT K:CLS
120 INPUT"introduza o número ch
ave (7 dígitos)";N$
130 FOR K=1 TO 1500:NEXT K:CLS
140 FOR K=1 TO LEN(A$)
150 L=K-INT(K/7)*7+1
160 T=ASC(MID$(A$,K,1))+(S*VAL(
MID$(N$,L,1)))
170 IF T>90 OR T<65 THEN T=T-(S
*26)
180 PRINT CHR$(T);
190 NEXT
```

```
20  HOME
30  PRINT  TAB( 11)"CIFRA SGT.
CYR": PRINT
40  PRINT  TAB( 14)"CUIDADO": P
RINT
50  PRINT "NAO DEIXE ESPACOS EN
TRE AS PALAVRAS"
60  PRINT : PRINT
70  PRINT "INTRODUZA  1  PARA C
ODIFICAR"
80  PRINT "INTRODUZA -1 PARA DE
CODIFICAR"
90  INPUT S
100  INPUT "QUAL SUA MENSAGEM?"
;A$
110  FOR K = 1 TO 1500: NEXT K:
 HOME
120  INPUT "INTRODUZA O NUMERO
CHAVE (7 DIGITOS)";N$
130  FOR K = 1 TO 1500: NEXT K:
 HOME
140  FOR K = 1 TO  LEN (A$)
150 L = K -  INT (K / 7) * 7 +
1
160 T =  ASC ( MID$ (A$,K,1)) +
 (S *  VAL ( MID$ (N$,L,1)))
170  IF T > 90 OR T < 65 THEN T
 = T - (S * 26)
180  PRINT  CHR$ (T);
190  NEXT
```

Cada letra lida do texto do programa é, antes, convertida em seu valor ASCII. A função **VAL** acrescenta a esse valor uma quantidade indicada por um dos dígitos de seu número-chave (linha 160). Depois de conferirmos se o resultado está na faixa aceitável (linha 170) a função **CHR$** apresentará a letra equivalente à original.

Consideremos a mensagem TROPASCAPTURAMPONTEPEGASUS. Rodando o programa com o número-chave 5331401, o texto codificado seria: WUPTATHDSUYRBRSROXEQJJDTYS. Com um indicador variável **S**, que pode assumir os valores −1 ou 1, é possível usar o mesmo programa para a decodificação.



O número-chave de sete dígitos pode ser escolhido aleatoriamente. Porém, como é necessário que ele esteja sempre disponível, é aconselhável utilizar algo que nos seja familiar, como um número de telefone. Como há dez milhões de combinações possíveis para o número-chave, a cifra de Saint Cyr dificilmente pode ser quebrada. Contudo, é possível torná-la ainda mais segura se codificarmos a mensagem duas vezes, isto é, colocando a mensagem já cifrada outra vez no computador.

Usando os números-chave 8694153 e 8130337, a seqüência de codificação e decodificação seria:

Texto	Número-chave	
PREPARARDESEMBARQUE		↑ CODIFICAÇÃO
	8694153	
VAIQFUIXMITJPJGAUVJ		↓ ↑ DECODIFICAÇÃO
	8130337	
WDITFBQYPIWMWRHDUYM		
	8130337	
VAIQFUIXMITJPJGAUVJ		
	8694153	↓
PREPARARDESEMBARQUE		

CÓDIGO MORSE

A segurança, porém, não é tudo. Por mais indecifrável que seja um código, ele de nada vale se não pode ser transmitido rapidamente.

O imperador francês Napoleão Bonaparte percebeu o problema e tentou superá-lo construindo por toda a França torres que se comunicavam por meio de sinais de luz. Isso tornou possível o envio de mensagens a grandes distâncias em pouco tempo. Dessa forma, o Exército francês podia manter-se informado sobre o movimento das tropas inimigas.

Entretanto, a invenção do telégrafo e da cifra conhecida como código Morse, na década de 1830, foram os fatores que mais contribuíram para dinamizar as comunicações secretas. O inventor desse código, o norte-americano Samuel Morse, criou um sistema de comunicação no qual pontos e traços substituiam as letras.

O programa *Código Morse* codificará seus textos e decodificará uma série de pontos e traços. Por exemplo, o famoso sinal de socorro SOS seria repre-

Qual a utilidade dos códigos?

Indispensáveis ao mundo dos espiões, os códigos são também muito empregados nas telecomunicações e em atividades bancárias que exigem sigilo. Grandes empresas comerciais que fazem uso de computadores para controlar seus estoques recorrem igualmente a esse tipo de linguagem. Assim, cada item estocado é classificado com um código. Quando se trata de artigos importados, costuma-se utilizar o "código de barras", que consiste em quadrinhos com faixas verticais pretas e brancas. Essas "barras" são lidas por meio de um instrumento óptico. Quando um dos artigos é retirado, o computador registra a operação e informa a respeito do preço da mercadoria.

Existem também terminais de compras nas próprias lojas, onde o cliente apenas digita um código e o total a pagar. Instantaneamente, o computador transfere essa quantia da sua conta para a da loja. Como se pode ver, as notas e moedas tendem a desaparecer, e as caixas registradoras deverão ser substituídas por terminais de computador. Mas, para tanto, é necessário que sejam criados códigos muito eficientes, de modo a evitar roubos e fraudes.

sentado por /.../---/.../ e um texto maior como FUJA À MEIA-NOITE seria /..-./..-/.---/.-/.-/--/./../.-/-./---/../-/./

UM EXERCÍCIO DE CODIFICAÇÃO

Tente fazer as duas codificações mencionadas com o próximo programa. Para os micros da linha Sinclair, você deve adicionar cinco asteriscos no final da mensagem para indicar que ela já está completa.

```
10 BORDER 0: PAPER 0: INK 7:
CLS : DIM a$(26,4): LET S$="
": LET f$="      "
15 POKE 23658,8
20 FOR x=1 TO 26: READ a$(x):
 NEXT x
30 INPUT "Codificar (1) ou De
codificar (2)";r
40 IF r=2 THEN  GOTO 140
60 INPUT "MENSAGEM A SER CODI
FICADA :"'m$
70 FOR x=1 TO LEN m$
80 IF m$(x)=" " THEN  PRINT "
    ";: GOTO 110
90 LET p$=m$(x)
100 PRINT "    ";a$((CODE p$)-
64);
110 NEXT x
120 PRINT '''" QUALQUER TECLA
 PARA CONTINUAR";: PAUSE 9999
130 RUN
140 INPUT "MENSAGEM A SER DECO
DIFICADA :"'m$: LET m$=m$+" "
160 FOR x=1 TO LEN m$
170 LET k$=m$(x)
180 IF k$=" " THEN  GOTO 220
190 LET s$=s$+k$
200 NEXT x: GOTO 120
210 IF LEN s$>5 OR LEN s$<1
THEN  PRINT "ERRO": GOTO 120
220 IF LEN s$<>5 THEN  LET s$=
s$+f$( TO 5-LEN s$)
225 FOR h=1 TO 26: IF a$(h)=s$
THEN  PRINT CHR$ (h+64);
230 NEXT h
240 LET s$="": GOTO 200
250 DATA "0-","-000","-0-0","-
00","0","00-0","--0","0000","0
0","0---","-0-","0-00","--","-
0","---","0--0","--0-","0-0","
000","-","00-","000-","0--","-
00-","-0--","--00"
```

```
10 CLS:DIM A$(26)
20 FOR X=1 TO 26:READ A$(X):NEX
T X
30 PRINT @96,;:INPUT"CODIFICAR
OU DECODIFICAR (1,2) ";R
40 IF R=2 THEN 140
50 PRINT"  MENSAGEM A SER CODIF
ICADA  :"
60 INPUT M$
70 FOR X=1 TO LEN(M$)
80 IF MID$(M$,X,1)=" " THEN PRI
NT"      ";:GOTO 110
90 P$=MID$(M$,X,1)
100 PRINT " ";A$(ASC(P$)-64);
110 NEXT
120 PRINT:PRINT:PRINT "  QUALQU
ER TECLA PARA CONTINUAR"
130 IF INKEY$="" THEN 130 ELSE
RUN
140 PRINT" MENSAGEM A SER DECOD
IFICADA"
150 INPUT M$:M$=M$+" ":PRINT:PR
INT
160 FOR X=1 TO LEN(M$)
170 K$=MID$(M$,X,1)
180 IF K$=" " THEN 210
190 S$=S$+K$
200 NEXT:PRINT:GOTO 120
210 IF LEN (S$)>4 OR LEN(S$)<1
THEN PRINT " ";:GOTO 200
220 FOR H=1 TO 26:IF A$(H)=S$ T
HEN PRINT CHR$(H+64);
230 NEXT
240 S$="":GOTO 200
250 DATA 0-,-000,-0-0,-00,0,00-
0,--0,0000,00,0---,-0-,0-00,--,
-0,---,0--0,--0-,0-0,000,-,00-,
000-,0--,-00-,-0--,--00
```

```
10 CLS:DIM A$(26)
20 FOR X=1 TO 26 :READ A$(X):NE
XT X
30 PRINT TAB(3);:INPUT"CODIFICA
```

```
R OU DECODIFICAR(1,2)";R
40 IF R=2 THEN 140
50 PRINT TAB(7)"MENSAGEM A SER
CODIFICADA"
60 INPUT M$
70 FOR X=1 TO LEN (M$)
80 IF MID$(M$,X,1)=" " THEN PRI
NT"     ";:GOTO 110
90 P$=MID$(M$,X,1)
100 PRINT" ";A$(ASC(P$)-64);
110 NEXT X
120 PRINT:PRINT:PRINT TAB(2)"AP
ERTE QUALQUER TECLA PARA CONTIN
UAR"
130 IF INKEY$="" THEN 130 ELSE
RUN
140 PRINT TAB(6)"MENSAGEM A SER
 DECODIFICADA"
150 INPUT M$:M$=M$+" ":PRINT:PR
INT
160 FOR X=1 TO LEN(M$)
170 K$=MID$(M$,X,1)
180 IF K$=" " THEN GOTO 210
190 S$=S$+K$
200 NEXT X:PRINT:GOTO 120
210 IF LEN(S$)>4 OR LEN(S$)<1 T
HEN PRINT:PRINT "    DEIXE ESPA
ÇO ENTRE OS CÓDIGOS": GOTO 200
220 FOR H=1 TO 26:IF A$(H)=S$ T
HEN PRINT CHR$(H+64);
230 NEXT
240 S$="":GOTO 200
250 DATA 0-,-000,-0-0,-00,0,00-
0,--0,0000,00,0---,-0-,0-00,--,
-0,---,0--0,--0-,0-0,000,-,00-,
000-,0--,-00-,-0--,--00
```

4

Código Morse na tela

```
10  HOME : DIM A$(26)
20  FOR X = 1 TO 26: READ A$(X)
: NEXT X
30  PRINT  TAB( 6);: INPUT "COD
IFICAR OU DECODIFICAR(1,2)?";R
40  IF R = 2 THEN 140
50  PRINT  TAB( 9)"MENSAGEM A S
ER CODIFICADA"
60  INPUT M$
70  FOR X = 1 TO  LEN (M$)
80  IF  MID$ (M$,X,1) = " " THE
N  PRINT "     ";: GOTO 110
90 P$ =  MID$ (M$,X,1)
100  PRINT " ";A$( ASC (P$) - 6
4);
110  NEXT
120  PRINT : PRINT : PRINT  TAB
( 3)"APERTE QUALQUER TECLA PARA
 CONTINUAR"
130  GET T$: IF T$ = "" THEN  G
OTO 130
135  GOTO 30
140  PRINT  TAB( 8)"MENSAGEM A
SER DECODIFICADA"
150  INPUT M$:M$ = M$ + " ": PR
INT : PRINT
160  FOR X = 1 TO  LEN (M$)
170 K$ =  MID$ (M$,X,1)
180  IF K$ = " " THEN  GOTO 210
190 S$ = S$ + K$
200  NEXT X: PRINT : GOTO 120
210  IF  LEN (S$) > 4 OR  LEN (
S$) < 1 THEN  PRINT " ";: GOTO
200
220  FOR H = 1 TO 26: IF A$(H)
= S$ THEN  PRINT  CHR$ (H + 64)
;
230  NEXT
240 S$ = "": GOTO 200
250  DATA  0-,-000,-0-0,-00,0,0
0-0,--0,0000,00,0---,-0-,0-00,-
-,-0,---,0--0,--0-,0-0,000,-,00
-,000-,0--,-00-,-0--,--00
```

A primeira parte do programa atribui a cada variável **A$(X)** um sinal Morse equivalente a uma letra do alfabeto, na sua devida ordem (linha 20). Por exemplo, **A$(1)** conterá /.-/, que é o código da letra A. Seria interessante usar o sinal de menos para um traço, e o asterisco ou o zero para um ponto, em vez do ponto final.

A parte de codificação é muito semelhante à dos dois primeiros programas. Cada letra de seu texto é lida e convertida em um número entre 1 e 26 e a seqüência adequada de pontos e traços é então impressa (linha 100).

A decodificação da mensagem se processa do seguinte modo. O computador lê cada caractere e soma-o aos anteriores, até encontrar um espaço (linhas 160 a 200). Então ele compara essa cadeia de caracteres com as que estão na variável **A$**, identificando a letra equivalente e imprimindo-a por meio do comando **CHR$** (linha 220).

Na segunda parte deste artigo, você aprenderá a decodificar transposições e cifras e utilizar códigos multiplicativos e códigos comerciais.

ARMAZENAGEM DE NÚMEROS

Como você já sabe, o interpretador BASIC pode trabalhar com números fracionários. Aprenda agora a armazená-los na memória e obtenha resultados mais precisos.

Quando o resultado de uma operação numérica é muito grande ou, então, muito pequeno, o computador mostra, freqüentemente, na tela, um número com aparência meio estranha, tal como 2.34E12 ou 1.222E-9.

Essa maneira de representar números é conhecida como *notação científica*, ou *notação de engenharia*, e é usada automaticamente toda vez que o interpretador BASIC não consegue mostrar um valor numérico que caia dentro do limite de sete a oito algarismos, dependendo da marca do computador. Existem também meios, em BASIC, de mostrar ou imprimir os valores numéricos que se quiser, em notação científica.

Na notação científica (chamada ainda de *forma exponencial*), um número é composto por duas partes: a *mantissa*, que contém os algarismos significativos do número, geralmente com apenas um dígito à esquerda da vírgula; e o *expoente*, que consiste em uma potência de 10 pela qual a mantissa deverá ser multiplicada para obter o valor numérico a ser representado. Assim, por exemplo, os números aqui abordados significam, respectivamente:

$$2.34E12 = 2.34 \times 10^{12} = 2.340.000.000.000$$

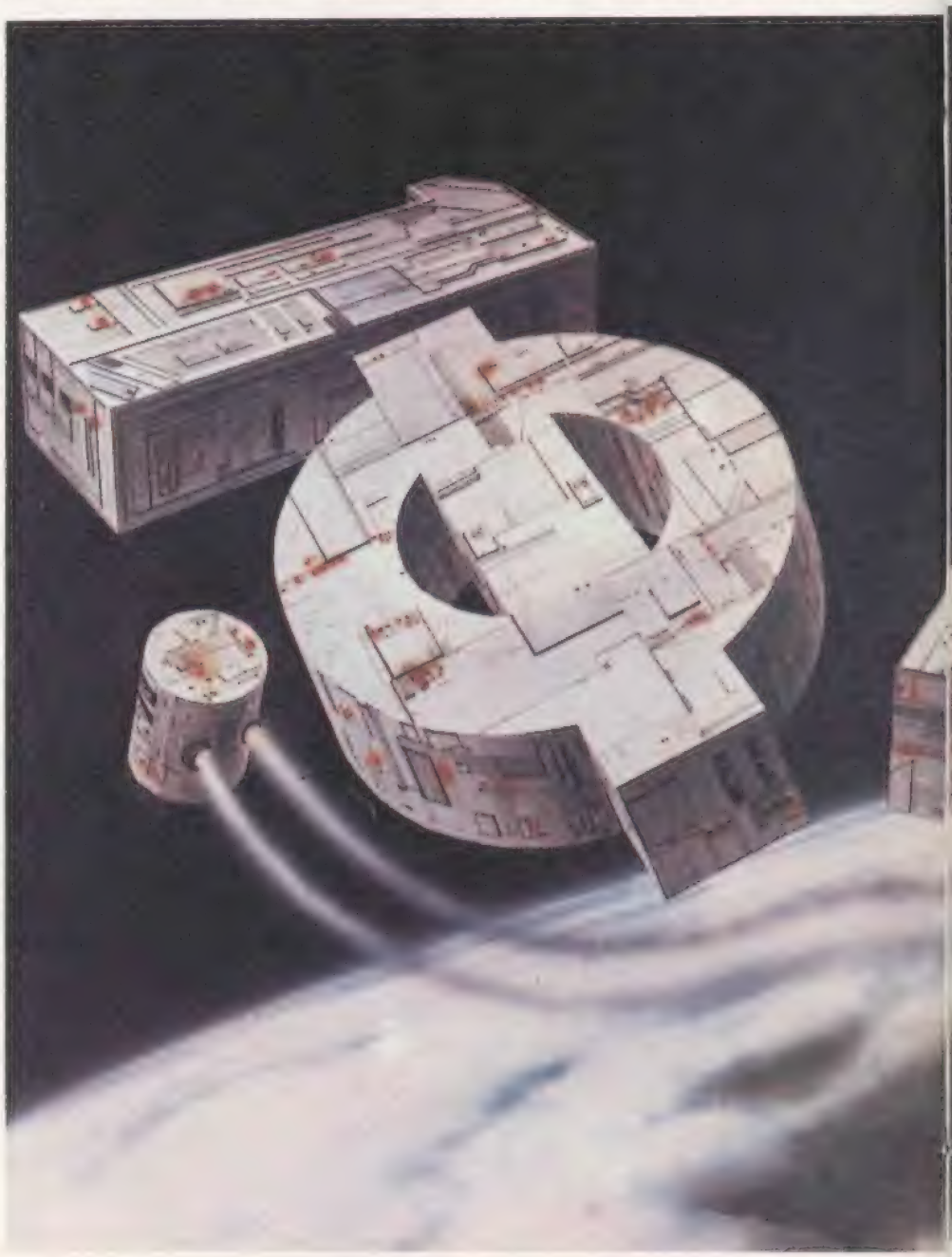
$$1.223E\text{-}9 = 1.223 \times 10^{-9} = 0{,}000000001223$$

Na realidade, esta é exatamente a maneira com que o interpretador BASIC armazena *todos* os valores reais: mantissa e expoente são conservados em grupos separados de bytes. Só quando o número vai ser mostrado por um **PRINT** ou equivalente é que se processa a conversão para o formato mais conhecido. O limite para que isso ocorra varia com a marca do computador. Nos micros da linha Apple, TRS-Color e TRS-80, por exemplo, é de sete algarismos. No Spectrum e no MSX, é de oito algarismos. Alguns computadores, como os das linhas TRS e MSX, podem expressar números com precisão dupla, isto é, não precisam recorrer à forma exponencial se o número tiver até quinze dígitos de precisão.

Você também poderá usar a forma exponencial como um modo abreviado de entrar números longos no computador (dentro de um programa, ou mesmo usando um comando **INPUT**).

O programa a seguir o ajudará a entender como funciona essa forma, e como o valor da mantissa é calculado.

```
10 CLS : POKE 23658,8
20 INPUT "INTRODUZA O NUMERO"
, LINE A$
25 IF A$="" THEN  GOTO 20
30 LET N$="0": LET E=0: LET N
```



- EXPOENTES
- COMO SÃO ARMAZENADOS OS NÚMEROS
- PONTO FLUTUANTE
- A FUNÇÃO INSTR
- NÚMEROS NEGATIVOS
- PEEK NA MEMÓRIA
- COMO ECONOMIZAR MEMÓRIA
- EFEITOS ESTRANHOS
- FORMATAÇÃO COM PRINT USING

```
-VAL A$
40 IF N=0 THEN  PRINT "VALOR
MUITO PEQUENO !": PAUSE 50:
GOTO 10
50 LET F=0: FOR M=1 TO LEN A$
: IF A$(M)="E" THEN  LET F=M
55 NEXT M: IF F=0 THEN  GOTO
 200
```

```
 60 LET N$=STR$ N: LET F=0:
 FOR M=1 TO LEN N$: IF N$(M)="
 E" THEN  LET F=M
 65 NEXT M: IF F=0 THEN  GOTO
  180
 68 LET N$=N$( TO F-1)
 70 IF ABS N<1 THEN  GOTO 130
 80 IF ABS N<10 THEN  GOTO 110
 90 LET N=N/10: LET F=0: FOR M
 =1 TO LEN N$: IF N$(M)="."
 THEN  LET F=M
 92 NEXT M: IF F=LEN N$ THEN
 LET N$=N$( TO LEN N$-1): GOTO
  100
 95 IF F<>0 THEN  LET N$=N$(
 TO F-1)+N$(F+1)+"."+N$(F+2 TO
 ): GOTO 80
100 LET N$=N$+"0": GOTO 80
110 IF N$(LEN N$)="." THEN
LET N$=N$( TO LEN N$-1)
120 GOTO 180
130 IF ABS N>=1 THEN  GOTO 170
140 LET N=N*10: LET F=0: FOR M
=1 TO LEN N$: IF N$(M)="."
THEN  LET F=M
145 NEXT M: IF F=1 THEN  LET N
$=".0"+N$(2 TO ): GOTO 130
150 IF F<>0 THEN  LET N$=N$(
TO F-2)+"."+N$(F-1)+N$(F+1 TO
): GOTO 130
160 LET N$="."+N$: GOTO 130
170 IF VAL A$<0 THEN  LET N$="
-"+N$
180 PRINT : PRINT A$;" IGUAL":
 PRINT N$
190 GOTO 20
200 IF ABS N<1 THEN  GOTO 220
210 IF ABS N>=10 THEN  LET E=E
+1: LET N=N/10: GOTO 210
215 GOTO 230
220 IF ABS N<=1 THEN  LET E=E-
1: LET N=N*10: GOTO 220
230 PRINT A$;" IGUAL": PRINT N
;: IF E<>0 THEN  PRINT "E";E
240 PRINT : GOTO 20
```

```
10 CLS
20 PRINT:INPUT"INTRODUZA O NUME
 RO ";A$
  30 N$="0":E=0:N=VAL(A$)
   40 IF N=0 THEN PRINT "VALOR
   MUITO PEQUENO!":PRINT:GOTO 20
    50 I=INSTR(A$,"E"):IF F=0
     THEN 200
     60 N$=STR$(N):F=INSTR(N$,"E
     "):I  F F=0 THEN 180 ELSE N
      $=MID$(N$,2,F-2)
    70 IF ABS(N)<1 THEN 130
80 IF ABS(N)<10 THEN 110
90 N=N/10:F=INSTR(N$,"."):IF F=
```

```
LEN(N$) THEN N$=LEFT$(N$,LEN(N$
)-1) ELSE IF F<>0 THEN N$=LEFT$
(N$,F-1)+MID$(N$,F+1,1)+"."+MID
$(N$,F+2):GOTO 80
100 N$=N$+"0":GOTO 80
110 IF RIGHT$(N$,1)="." THEN N$
=LEFT$(N$,LEN(N$)-1)
120 GOTO 180
130 IF ABS(N)>=1 THEN 170
140 N=N*10:F=INSTR(N$,"."):IF F
=1 THEN N$=".0"+MID$(N$,2):GOTO
 130
150 IF F<>0 THEN N$=LEFT$(N$,F-
2)+"."+MID$(N$,F-1,1)+MID$(N$,F
+1):GOTO 130
160 N$="."+N$:GOTO 130
170 IF VAL(A$)<0 THEN N$="-"+N$
180 PRINT:PRINT A$;" IGUAL":PRI
NT N$
190 GOTO 20
200 IF ABS(N)<1 THEN 220
210 IF ABS(N)>=10 THEN E=E+1:N=
N/10:GOTO 210 ELSE 230
220 IF ABS(N)<=1 THEN E=E-1:N=N
*10:GOTO 220
230 PRINT A$;" IGUAL ":PRINT N;
CHR$(8);:IF E<>0 THEN PRINT "E"
;E
240 GOTO 20
```

```
10  HOME
20  PRINT : INPUT "ENTRE NUMERO
:";A$
30  LET N$ = "0":E = 0:N =  VAL
 (A$)
40  IF N = 0 THEN  PRINT "VALOR
 MUITO PEQUENO": PRINT : GOTO 2
0
50  FOR F = 1 TO  LEN (A$): IF
 MID$ (A$,F,1) = "E" THEN 54
52  NEXT F:F = 0: GOTO 200
54  IF F = 0 THEN 200
60  LET N$ =  STR$ (N)
62  FOR F = 1 TO  LEN (N$): IF
 MID$ (N$,F,1) = "E" THEN 66
64  NEXT F:F = 0
66  IF F = 0 THEN 180
68  LET N$ =  MID$ (N$,1,F - 2)
70  IF  ABS (N) < 1 THEN 130
80  IF  ABS (N) < 10 THEN 110
90  LET N = N / 10
92  FOR F = 1 TO  LEN (N$): IF
 MID$ (N$,F,1) = "." THEN 96
94  NEXT F:F = 0
96  IF F =  LEN (N$) THEN N$ =
 LEFT$ (N$, LEN (N$) - 1): GOTO
 100
98  IF F <  > 0 THEN N$ =  LEFT
$ (N$,F - 1) +  MID$ (N$,F + 1,
1) + "." +  MID$ (N$,F + 2): GO
```

```
TO 80
100  LET N$ = N$ + "0": GOTO 80
110  IF  RIGHT$ (N$,1) = "." TH
EN N$ =  LEFT$ (N$, LEN (N$) -
1)
120  GOTO 180
130  IF  ABS (N) >  = 1 THEN 17
0
140  LET N = N * 10
142  FOR F = 1 TO  LEN (N$): IF
  MID$ (N$,F,1) = "." THEN 146
144  NEXT F:F = 0
146  IF F = 1 THEN N$ = "0.0" +
  MID$ (N$,2): GOTO 130
150  IF F <  > 0 THEN N$ =  LEF
T$ (N$,F - 2) + "." +  MID$ (N$
 F - 1,1) +  MID$ (N$,F + 1): G
OTO 130
160  LET N$ = "." + N$: GOTO 13
0
170  IF  VAL (A$) < 0 THEN N$ =
 "-" + N$
180  PRINT A$;" IGUAL A ";N$
190  GOTO 20
200  IF  ABS (N) < 1 THEN 220
210  IF  ABS (N) >  = 10 THEN E
 = E + 1:N = N / 10: GOTO 210
215  GOTO 230
220  IF  ABS (N) <  = 1 THEN E
 = E - 1:N = N * 10: GOTO 220
230  PRINT A$;" IGUAL A ";N;
235  IF E <  > 0 THEN  PRINT "E
";E
240  PRINT : GOTO 20
```

Quando você executar esse programa, com o comando **RUN**, o computador ficará esperando que um número seja digitado, e as teclas <**ENTER**> ou <**RETURN**> sejam pressionadas. Entre, então, um número positivo, em forma normal, ou no modo exponencial. O computador imprimirá o número que você entrou na forma oposta. Os micros TRS-80, TRS-Color e MSX dispõem de um comando (**PRINT USING**, já explicado em um artigo anterior) que dá mais liberdade ao programador para imprimir números de acordo com um determinado formato. Isso será explicado mais adiante.

O programa utiliza a função **INSTR**, nas versões para o TRS-80, TRS-Color e MSX. Essa função não existe no Apple e no Spectrum, mas pode ser substituída por uma pequena rotina sempre que for necessário. Ela tem por finalidade localizar o ponto da cadeia que contém o número de entrada, onde está (se estiver) presente a letra E, indicativa de expoente. A expressão **INSTR (A$, "E")** retorna um valor numérico, igual à posição do E na cadeia de entrada. Esse valor é armazenado em **F**, e será igual a zero, se não for encontrado.

Nos computadores que não possuem a função **INSTR**, um laço **FOR...NEXT** percorre, caractere por caractere, a cadeia de entrada (função **MID$**, no Apple, e **TO**, no Spectrum), examinando se é um E. Se este for encontrado, sua posição estará contida no contador do laço (variável **F**).

A lógica do programa é um pouco complicada, mas você conseguirá entendê-la se tomar um exemplo numérico e seguir o processamento manualmente, para ver como funciona.

COMO CONVERTER

Se você quiser encontrar a forma "normal" de um número expresso na forma exponencial, deve multiplicar a mantissa por um valor igual a 10 elevado ao número situado logo após a letra E (ou seja, o expoente).

Aparentemente complicado, esse cálculo é na verdade muito simples. Tome como exemplo o número 1.23E4. Para convertê-lo na forma mais comum, multiplique a mantissa (1.23) por 10 elevado a 4. Teremos, assim:

$$1.23 \times 10.000 = 12.300$$

Você pode tentar converter vários números como esse, checando os resultados com o auxílio do programa que foi apresentado linhas atrás. Note que, qualquer que seja o valor, a mantissa é sempre expressa com um algarismo antes do ponto. Portanto, ela varia entre 1.0 e 9.999999.

Os valores negativos são expressos de maneira similar. Nesse caso, a mantissa é um número negativo, enquanto o expoente pode ser tanto negativo como positivo. Aliás, um sinal negativo no expoente tem um significado totalmente diferente do de um sinal negativo na mantissa. Tente você mesmo testar isso, utilizando nosso programa.

ARMAZENAMENTO DE NÚMEROS

A notação exponencial é igualmente útil para nos ajudar a compreender como os computadores armazenam números, internamente.

Quando um comando direto é digitado e executado — por exemplo, **PRINT 10 * 10** —, a primeira coisa que o interpretador BASIC faz é traduzir os números entrados para a forma exponencial, e usá-los nessa forma para calcular o resultado, que também é armazenado na forma exponencial.

Nem tudo, porém, parece tão simples. Como você já sabe, os computadores digitais não utilizam a notação decimal para armazenar números, mas sim a binária (numeração de base dois). Para entender de que maneira os números são armazenados na memória RAM, siga o exemplo abaixo, de conversão do número 10 (em decimal).

O primeiro passo é convertê-lo em binário. Isso nos dá o número

00001010.00000...

Os primeiros quatro zeros desse número binário poderiam ser eliminados, resultando no número 1010.00000...

Como foi visto, um número decimal em forma exponencial contém uma mantissa entre 1.0 e 9.9999... Quando um número como esse é armazenado em binário, a mantissa transforma-se em um número que sempre começa com .1.

Isso é possível porque o tamanho da parte exponencial do número determina a posição do *ponto binário* (o equivalente binário ao ponto decimal). Como ele é automaticamente definido, a mantissa não precisa especificar onde se encontra o ponto. Surgem, assim, dois problemas: como fazer com que a parte exponencial do número indique onde deverá estar o ponto binário, e como converter esse número de tal maneira que ele comece com .1?

COMO MOVIMENTAR O PONTO

A resposta para ambos os problemas é a mesma: tudo o que precisa ser feito

é mover o ponto até que ele fique à esquerda do primeiro 1, e adicionar 1 ao expoente para cada posição que o ponto seja deslocado. Por exemplo, para o número 58 (em decimal), ou 111010.000 (em binário), o ponto binário é movido gradualmente para a esquerda, até que não haja mais números à sua esquerda. Neste caso, teríamos que movê-lo seis casas para a esquerda, de modo que o expoente resultasse em +6, e a mantissa fosse 0.11101000.

Na realidade, o expoente parte de 128 (por razões esclarecidas mais adiante); portanto, o número de posições deslocadas deve ser somado a 128, e não a 0. No exemplo acima, o expoente será igual a 128 + 6, ou 134 (e isto é armazenado como um número binário).

Recapitulando, o computador armazena a parte exponencial de um número em byte. A mantissa é armazenada também, só que ocupa quatro bytes. No exemplo acima, os três bytes restantes seriam preenchidos com zeros:

EXPOENTE

MANT-1 MANT-2 MANT-3 MANT-4

11101000 00000000 00000000 00000000

Portanto, qualquer número real ocupa exatamente cinco bytes. Isso limita os tamanhos máximo e mínimo que o computador pode armazenar. O byte único da parte exponencial nos dá o valor máximo de 2 elevado a 127 (não de 256, pois o primeiro bit é usado como indicador de sinal, para comunicar se o expoente é positivo ou negativo, deixando apenas sete bits para o valor numérico do expoente).

O valor máximo da mantissa é 0.11111..., quando todos os bits são iguais a 1. Esse valor está muito próximo do binário 1.00000..., que é 1 tanto em binário quanto em decimal. O menor valor possível será, então, 0.100000..., quando todos os bits, exceto o primeiro, forem iguais a zero (lembre-se de que o primeiro bit da mantissa é sempre igual a 1, pois o ponto binário é movido até aí). E 0.1 em binário é igual a 1/2 ou 0.5, em decimal.

Multiplicando a mantissa pelo expoente, você poderá chegar até os valores extremos que podem ser armazenados na memória, em ponto flutuante. O valor máximo é 1.70141E38, em decimal, para um micro de oito bits. Você poderá utilizar os programas seguintes para verificar como é representado este número internamente.

Nesse aspecto, o Spectrum difere dos demais computadores: ele utiliza seis bytes em vez de cinco para armazenar números em ponto flutuante. Cinco desses bytes são idênticos aos do exemplo citado. O sexto é necessário para indicar ao computador se o número é de ponto flutuante ou não. Isso é feito precedendo-se o grupo de cinco bytes com o byte igual a 14.

CASOS ESPECIAIS

O que foi dito anteriormente nos dá uma boa idéia de como o computador armazena números em formato de ponto flutuante. Existem, porém, duas variantes em relação ao processo estudado. A primeira ocorre quando o número inicial é menor do que 1. Nesse caso, se tentarmos movimentar o ponto binário para a esquerda, iremos afastá-lo do lugar onde queremos que ele realmente esteja. Portanto, devemos movê-lo para a direita. Do mesmo modo, a fim de que o número seja corretamente representado, diminui-se o expoente de 1, para cada posição deslocada (em vez de somar 1). O expoente começa, como sempre, a partir do valor 128.

Muitas vezes, antes de exibir um número binário, o computador escreve, acima de cada "coluna" desse número, seu equivalente decimal. Você já deve ter visto isso antes nos artigos sobre blocos gráficos, mas apenas para o caso dos números à esquerda do ponto. O mesmo pode ser feito para os números à direita do ponto, que formam na realidade sua parte fracionária.

Em binário, para achar o valor decimal da próxima coluna à esquerda, multiplica-se o valor por 2. Trabalhando da esquerda para a direita, portanto, deve-se fazer exatamente o contrário, ou seja, dividir por 2.

A segunda variante na representação dos números em ponto flutuante ocorre quando o número é menor do que zero, ou seja, negativo. Nesse caso, o computador precisa armazenar a sua versão interna do sinal de menos em algum lugar — e isto deve ser feito sem desperdiçar espaço de memória.

Você viu antes que, da maneira como os números são armazenados, o primeiro bit do primeiro byte da mantissa tem que ser sempre igual a 1. Dando is-

so como certo, o computador utiliza esse bit para indicar o sinal da mantissa: se o número for negativo, o bit valerá 1; se ele for positivo, o valor do bit será, então, 0.

ONDE ESTÁ AQUELE NÚMERO?

A partir dos exemplos acima, já é possível entender como o computador armazena os números na memória. O número decimal 58 (111010 em binário) é armazenado em cinco bytes sucessivos, da maneira como se segue:

134 104 000 000 000 (em decimal)

e o número 10 (decimal) é armazenado assim:

132 032 000 000 000

Experimente vários exemplos, trabalhando com os valores decimais contidos nos bytes. Teste as suas respostas usando o programa a seguir. Digite **RUN 100** para entrar outro número.

Embora armazenem números na forma geral exposta até agora, os micros compatíveis com o Sinclair Spectrum contam ainda com outro recurso. Se um número inteiro, compreendido entre -65535 e 65535 for usado, o Spectrum o armazena de modo diferente, reduzindo-o a seu equivalente binário e guardando-o em apenas dois bytes (e não em forma exponencial, usando seis bytes). Embora possa parecer que isso seja feito para economizar memória, tal fato não acontece, pois três bytes são deixados sem uso.

Por essa razão, quando você usar o primeiro dos dois programas abaixo, não entre um valor numérico entre -65535 e 65535: o programa ainda funcionará, mas os resultados não corresponderão à explicação adotada.

```
10 BORDER 1: PAPER 7: INK 9:
CLS
20 INPUT "Introduza um numero
(nao inteiro)",x
40 PRINT '"Expoente: ";PEEK (
PEEK 23627+256*PEEK 23628+1)
50 PRINT '"Mantissa:";: FOR n
=2 TO 5
60 PRINT TAB 10;PEEK (PEEK
23627+256*PEEK 23628+n)
70 NEXT n
80 STOP
```

```
10 CLS
20 INPUT" INTRODUZA UM NUMERO "
;N
30 D=VARPTR(N)
40 PRINT:PRINT" EXPOENTE = ",PE
EK(D)
50 PRINT" MANTISSA = ";
60 FOR G=1 TO 4
70 PRINT,PEEK(D+G)
80 NEXT
90 PRINT:GOTO 20
```

```
10  HOME
20  INPUT "ENTRE UM NUMERO:";X
30  LET V =  PEEK (105) +  PEEK
 (106) * 256
40  PRINT "EXPOENTE:"; PEEK (V
+ 2)
50  PRINT "MANTISSA:";
60  FOR N = 3 TO 6
70  PRINT  PEEK (V + N);" ";
80  NEXT N
```

```
100 REM SEGUNDO PROGRAMA
110 BORDER 1: PAPER 7: INK 9:
CLS : LET r=0
120 INPUT AT 1,0;"Introduza ex
poente",exp
130 IF exp<0 OR exp>255 THEN
GOTO 120
140 PRINT '"Expoente: ";exp:
POKE PEEK 23627+256*PEEK 23628
+1,exp
150 PRINT '"Mantissa:";: FOR n
=2 TO 5
160 INPUT AT 1,0;"Introduza ma
ntissa",man
170 IF man<0 OR man>255 THEN
GOTO 160
180 PRINT TAB 10;man: POKE
PEEK 23627+256*PEEK 23628+n,
man
190 NEXT n
200 PRINT '"Resulta: ";r
```

```
100 REM SEGUNDO PROGRAMA
110 CLS
120 N=1:D=VARPTR(N)
130 INPUT" INTRODUZA EXPOENTE "
;E
140 POKE D,(255 AND E)
150 PRINT" INTRODUZA MANTISSA "
;
160 FOR K=1 TO 4
170 INPUT E
180 POKE D+K,(255 AND E)
190 PRINT ,;
200 NEXT
210 PRINT:PRINT" NUMERO E ";N:P
RINT
220 GOTO 130
```

```
110  HOME :R = 1:V =  PEEK (105
) +  PEEK (106) * 256
120  INPUT "ENTRE EXPOENTE:";EX
130  IF EX < 0 OR EX > 255 THEN
 120
140  POKE V + 2,EX
150  FOR N = 3 TO 6
160  INPUT "ENTRE MANTISSA:";M
170  IF M < 0 OR M > 255 THEN 1
60
180  POKE V + N,M
190  NEXT N
200  PRINT "RESULTADO=";R
```

O primeiro desses programas permite que um número seja digitado pelo teclado; em seguida, ele imprime os conteúdos decimais dos cinco bytes onde o número está guardado.

Seu funcionamento tem a seguinte dinâmica: inicialmente, é necessário estabelecer uma variável numérica, que armazenará o número de entrada. Como ela é a primeira variável a ser definida quando o programa é executado, seu nome aparecerá logo no começo de uma área especial da memória (definida e fixada pelo interpretador BASIC de cada computador), chamada lista de variáveis. Essa lista tem informações sobre o nome da variável e o endereço da memória RAM onde seu conteúdo será armazenado. Tal endereço é dividido em dois bytes sucessivos. No Apple, por exemplo, o endereço do conteúdo da primeira variável de um programa encontra-se nos apontadores 105 e 106 (em decimal). No Spectrum, os endereços correspondentes são 23627 e 23628. Tanto no Apple quanto no Spectrum, o endereço decimal de dezesseis bits é calculado pela expressão na linha 40 (30 no Apple).

Nos computadores das linhas TRS-80, TRS-Color e MSX, o usuário não precisa conhecer a locação exata do apontador da lista de variáveis, pois existe uma função predefinida do BASIC chamada **VARPTR** (abreviatura de *VARiable PoinTeR* — ou apontador de variáveis), que indica o endereço decimal completo do ponto da memória onde está armazenado o conteúdo da variável nomeada no argumento da função. Veja a linha 30 do programa para esses computadores. A variável **D** conterá o endereço inicial dos cinco bytes que contém o valor armazenado na variável **N**. O primeiro byte contém o expoente e nos quatro bytes sucessivos está a mantissa.

O segundo programa faz o contrário: pede que cinco bytes sejam digitados, e imprime o número que ele representa (algumas das respostas poderão aparecer na forma exponencial).

Seu funcionamento é semelhante ao do primeiro programa, ou seja, comandos **PEEK** (nas versões para o Apple e Spectrum) ou **VARPTR** (nas versões para o TRS e o MSX) são usados para localizar o endereço inicial da primeira va-

riável. Ela é igualada a 0 ou a 1 logo na primeira linha do programa, para assegurar que ficará em primeiro lugar na lista de variáveis (esse "truque" só é necessário para o Apple e o Spectrum). A seguir, o programa pede ao usuário que entre os valores do expoente e das partes da mantissa. Em vez de empreender cálculos infindáveis para chegar ao resultado convertido, o programa coloca esses valores nos cinco bytes de armazenamento da variável inicial, por meio de comandos **POKE**, e imprime o resultado, por meio de um **PRINT** normal.

ECONOMIZE MEMÓRIA

O computador sempre armazena números em grupos de cinco bytes. Ora, isso representa um grande desperdício de memória, pois nem sempre são necessários tantos bytes na mantissa. Além disso, os bytes em excesso poderiam ser aproveitados em outros lugares.

De fato, você pode economizar quantidades significativas de memória se evitar o uso de *números* em BASIC, embora isso nem sempre seja possível. Existe, entretanto, um jeito de reduzir a quantidade de memória que cada número requer. Assim, é possível, em muitos casos, evitar o emprego de números em um programa, colocando dentro de variáveis os números repetidamente utilizados em várias partes do programa. Por exemplo, uma constante que aparece com freqüência , como o número *pi* (3.1415...), pode ser colocado na variável **P**. Esse "truque" funciona sempre, mas não vale a pena recorrer a ele quando o valor numérico é usado apenas uma ou duas vezes no programa.

A vantagem de se empregar uma variável nesses casos é que o seu nome ocupa apenas um ou dois bytes de espaço no programa, em vez de cinco, se seu nome for curto. (A vantagem deixará, naturalmente, de existir se a variável se chamar **MEDIA**, ou coisa parecida).

Alguns números — como 0, 1, 2, 10 etc. — aparecem tantas vezes na maioria dos programas, que talvez convenha colocá-los em variáveis, se você tem problema de espaço. Esse método é especialmente útil nos micros Sinclair, que usam seis bytes em vez de cinco.

EFEITOS ESTRANHOS

Cálculos realizados por computador revelam, às vezes, erros aparentemente inexplicáveis; assim, um resultado que deveria dar "redondo" — 10.000000, por exemplo —, pode aparecer na tela sob uma forma levemente modificada — como 10.000001. Conhecidos como *erros de precisão*, esses equívocos são facilmente explicados pela maneira como o computador armazena números.

Tente os exemplos seguintes em sua máquina. A causa por trás de cada um dos erros será explicada mais adiante:

Primeiro, tente a linha abaixo:

```
PRINT 10.0000000001
```

Ao imprimir o resultado, o computador omite o 1 final e converte o número que você digitou em 10, simplesmente. Isso pode se tornar um problema, quando você quiser trabalhar com maior precisão numérica. Entretanto, é possível reverter esse fato em seu favor, quando se deseja entrar números entre -65535 e 65535 no programa (o que normalmente não deve ser feito, pois o Spectrum os representa de forma diferente, como já foi dito).

Basta, para isso, entrar o número inteiro em forma fracionária e colocar um algarismo pouco significativo no final. Assim, entrando o número 10.0000000001, como no exemplo citado, você "enganará" o microcomputador e ele mostrará como 10 é armazenado no formato de cinco bytes, em vez de no formato especial para números inteiros.

Agora tente entrar esta linha:

```
PRINT INT -65536
```

Com ela, será evidenciado um erro no interpretador BASIC do Spectrum, uma vez que ele não consegue diferenciar entre esse valor e -1.

Você pode testemunhar os estranhos efeitos causados nos micros compatíveis com o TRS-Color, entrando os números abaixo no primeiro programa:

```
12345678901 E-4
```

e também:

```
454545454 E-46
```

A discrepância do primeiro número é bastante óbvia. No segundo, o último dígito é mudado pelo computador.

Essas imprecisões não são, contudo, como muitas pessoas imaginam, erros na ROM dos computadores (exceção feita ao segundo exemplo para o Spectrum). A razão pela qual o algarismo 1 aparece ou desaparece, quando colocado no final do número, decorre do fato de que os computadores digitais, pela sua própria natureza, trabalham com limites no grau de precisão. Portanto, quando os valores excederem um determinado número de decimais, i interpretador será obrigado a arredondar o resultado.

Quando isso acontece, dependendo dos números que estão sendo usados e, também, do número de etapas de cálculo que o computador terá que vencer, os resultados finais podem ser completamente distintos do que se espera.

Os computadores são mais do que precisos a maior parte das vezes, e esses "erros", na realidade, não têm grande influência. Mas, para aplicações mais sérias, e que exigem grande precisão, como programas de contabilidade, de estatística etc., as incorreções cumulativas podem se tornar um grande inconveniente; isso é levado em consideração pelos melhores pacotes de software.

Por exemplo, usando conjuntos e matrizes, podem-se armazenar números com mais decimais do que o máximo de sete a nove permitido pelos computadores descritos no artigo. A vantagem dessa abordagem é que o programador pode desenvolver uma rotina para imprimir os números em forma normal, em vez da exponencial, como o computador faria com valores extremos. Essa é uma maneira, também, de evitar mensagens de erro, como "*overflow*", "número muito grande" etc.

FORMATAÇÃO DE NÚMEROS

Já explicamos que alguns microcomputadores, como o TRS-80, o TRS-Color e o MSX, têm comandos que permitem mudar o formato de impressão de números com o **PRINT** ou o **LPRINT**.

Embora uma tela possa ser formatada muito bem com o auxílio dos comandos **PRINT@** (nos TRS) ou **LOCATE** (no MSX), é conveniente recorrer à declaração **USING** para formatação, que é sempre usada em conjunto com os comandos **PRINT** ou **LPRINT**. Os elementos essenciais dessa declaração já foram explicados em artigo anterior; assim, faremos aqui apenas uma nova exposição, por intermédio de um programa simples de demonstração financeira para uma companhia fictícia.

```
10 CLS
20 PRINT"  A DIRETORIA TEM O PR
AZER DE    ANUNCIAR O BALANCO P
```

```
RELIMINAR              DE 1986 DA
:"
30 PRINT:PRINT"   FABRICA ACME
DE ARRUELAS   "
40 PRINT:PRINT:PRINT"O MAIOR FO
RNECEDOR DO MUNDO DE":PRINT
50 C$(0)="MD.345 - AZUIS":C$(1)
="MD.897 - VERDES":C$(2)="MD.91
2 - AMARELAS":C$(3)="MD.989 - V
ERMELHAS"
60 FOR K=0 TO 3
65 X$="ARRUELAS "+MID$(C$(K),10
)
70 PRINT TAB(16-LEN(X$)/2);X$
80 NEXT
90 FOR K=1 TO 7000:NEXT:CLS
110 PRINT @12,"JANEIRO":PRINT @
44,"-------"
120 PRINT:PRINT"PRECO MEDIO (AT
ACADO)"
130 A$="  ¢    ¢":B$="        *
*$##.##"
140 PRINT:PRINTUSING A$;C$(0);:
PRINTUSING B$;12.715265
150 PRINTUSING A$;C$(1);:PRINTU
SING B$;3.7363141
160 PRINTUSING A$;C$(2);:PRINTU
SING B$;10.35824221
170 PRINTUSING A$;C$(3);:PRINTU
SING B$;.5163733
180 PRINT @416,USING"LUCRO TOTA
L     $###,###.##";374241.5353
```

Esse programa, tal como está, funcionará bem apenas no micro TRS-Color. Para adaptá-lo para o TRS-80, faça as seguintes alterações:

```
70 PRINT TAB(32-LEN(C$(K)));C$(
K)
90 FOR K= 1 TO 4000:NEXT:CLS
110 PRINT @12,"JANEIRO": PRINT
@76, "-------"
180 PRINT @832,USING"LUCRO BRU-
TO    $###,###.##";374241.5353
```

```
10 SCREEN1:WIDTH32:CLS
20 PRINT"  A DIRETORIA TEM O PR
AZER DE    ANUNCIAR O BALANCO P
RELIMINAR              DE 1986 DA
:"
30 PRINT:PRINT"   FABRICA ACME
DE ARRUELAS   "
40 PRINT:PRINT:PRINT"O MAIOR FO
RNECEDOR DO MUNDO DE":PRINT
50 C$(0)="MD.345 - AZUIS":C$(1)
="MD.897 - VERDES":C$(2)="MD.91
2 - AMARELAS":C$(3)="MD.989 - V
ERMELHAS"
60 FOR K=0 TO 3
65 X$="ARRUELAS "+MID$(C$(K),10
)
70 PRINT TAB(16-LEN(X$)/2);X$
80 NEXT
90 FOR K=1 TO 7000:NEXT:CLS
110 PRINT TAB(12);"JANEIRO":PRI
NT TAB(12);"-------"
120 PRINT:PRINT"PRECO MEDIO (AT
ACADO)"
130 A$="  /    /":B$="        *
*$##.##"
140 PRINT:PRINTUSING A$;C$(0);:
PRINTUSING B$;12.715265
150 PRINTUSING A$;C$(1);:PRINTU
SING B$;3.7363141
160 PRINTUSING A$;C$(2);:PRINTU
SING B$;10.35824221
170 PRINTUSING A$;C$(3);:PRINTU
SING B$;.5163733
180 PRINT @416,USING"LUCRO TOTA
L     $###,###.##";374241.5353
```

As linhas 10 a 90 exibem na tela o título do relatório, formatado de maneira simples com declarações **PRINT**. A linha 170 é particularmente interessante porque mostra como uma série de mensagens pode ser centrada na tela, usando-se **TAB**, que é calculado de acordo com o comprimento da mensagem.

A seção do programa que vai da linha 100 à linha 180 demonstra as características da declaração do **PRINT USING**. A linha 130 define **A$**, que pode ser usada para mostrar as primeiras seis letras das cadeias alfanuméricas definidas na linha 50. O comprimento da cadeia é o número de espaços entre sinais % (no MSX o sinal usado é a barra inversa), mais 2. Assim, o comprimento do cordão de saída levará em conta o espaço ocupado pelos sinais %.

B$ define o formato da saída numérica. A função principal da formatação numérica é alinhar uma coluna de números, padronizando-os para que tenham a mesma quantidade de algarismos antes e depois do ponto, que sejam tabulados tomando o ponto como referência, e assim por diante.

O número de dígitos a ser mostrado é definido pelo símbolo #. Inserindo-se o ponto decimal na cadeia de # define-se a sua posição. Examine a linha 130 do programa: **B$** serve para imprimir números com dois dígitos antes do ponto e dois dígitos depois.

Além disso, se o sinal $ for colocado à esquerda dos sinais #, ele será impresso nessa posição em todas as saídas, e será usado para indicar quantias monetárias.

O símbolo **antes dos sinais $ e # fará com que os espaços em branco à esquerda do campo definido para o valor numérico seja preenchido automaticamente por asteriscos. Este é um recurso usado no preenchimento de cheques. No campo de saída definido na linha 130 foram empregados quatro sinais diferentes. Outras combinações são possíveis.

A linha 140 imprimirá os seis primeiros caracteres de **C$(0)**, seguidos por oito espaços e um sinal de $. Note que a declaração **USING** utiliza a cadeia de formatação armazenada previamente em **B$**. O número a ser mostrado nesse exemplo será, portanto, arredondado, de modo a aparecer apenas com duas casas depois do ponto.

A linha 150 imprimirá os seis primeiros caracteres de A$, seguidos de *$3.74, pois sobrou um espapo em branco na formatação, que será preenchido por um asterisco. A linha 160 é similar à linha 140. Finalmente, a linha 170 imprimirá o número como *$0.52.

Se os números a imprimir forem muito grandes, pode-se recorrer a três tipos de formatação. A mais simples consiste na utilização de uma cadeia longa de sinais #. Por outro lado, para visualizá-la melhor, podemos empregar vírgulas para separar os dígitos em grupos de três (no Brasil usaríamos pontos, mas os interpretadores BASIC existentes em nossas máquinas seguem o padrão norte-americano). Isso foi feito na linha 180 do programa. A terceira alternativa é recorrer à forma exponencial. Para ver como ela funciona, substitua a linha 180 por:

```
180 PRINT @832,USING"LUCRO BRU-
TO    $##.##^^^^";374241.5353
```

Faça a modificação pertinente no programa para o MSX. Os quatro sinais de ↑ que aparecem no formato indicam o tamanho do campo do expoente. Os sinais # são interpretados, então, como o formato da mantissa.

A declaração **USING** pode ainda ser misturada com um **PRINT@** na mesma linha (computadores TRS-80 e TRS-Color) e várias **USING** podem ser colocadas na mesma linha. Isto dá grande flexibilidade na programação de telas.

Outro caractere de programação de formato usado com a **USING** e ao qual não nos referimos antes é o ponto de exclamação (!). Ele faz com que o programa imprima apenas o primeiro caractere de uma cadeia alfanumérica. Tente alterar a linha 130 do programa:

```
130 A$="  !":B$="
**$##.##"
```

Alguns programas utilizam a função **VAL** seguida de um número entre aspas, em vez de um número normal. Essa forma serve, muitas vezes, para economizar espaço. O computador normalmente recorre a bytes de memória para armazenar um número, o que o leva a desperdiçar memória em certas aplicações críticas. Assim, se você puser um número entre aspas e usar a função **VAL** para convertê-lo em seu valor numérico, poderá armazená-lo como se fosse um cordão. No caso de números curtos (ou seja, com menos de cinco dígitos), esse procedimento levará a uma economia de memória: um byte para cada dígito, um para cada aspa, e um para a função **VAL**.

CLASSIFICAÇÃO DOS MICROCOMPUTADORES POR LINHA

LINHA	FABRICANTE	MODELO
Apple II +	Appletronica	Thor 2010
Apple II +	CCE	MC-4000 Exato
Apple II +	CPA	Absolutus
Apple II +	CPA	Polaris
Apple II +	Digitus	DGT-AP
Apple II +	Dismac	D-8100
Apple II +	ENIAC	ENIAC II
Apple II +	Franklin	Franklin
Apple II +	Houston	Houston AP
Apple II +	Magnex	DM II
Apple II +	Maxitronica	MX-2001
Apple II +	Maxitronica	MX-48
Apple II +	Maxitronica	MX-64
Apple II +	Maxitronica	Maxitronic I
Apple II +	Microcraft	Craf II Plus
Apple II +	Milmar	Apple II Plus
Apple II +	Milmar	Apple Master
Apple II +	Milmar	Apple Senior
Apple II +	Omega	MC-400
Apple II +	Polymax	Maxxi
Apple II +	Polymax	Poly Plus
Apple II +	Spectrum	Microengenho I
Apple II +	Spectrum	Spectrum ed
Apple II +	Suporte	Venus II
Apple II +	Sycomig	SIC I
Apple II +	Unitron	AP II
Apple II +	Victor do Brasil	Elppa II Plus
Apple II +	Victor do Brasil	Elppa Jr.
Apple IIe	Microcraft	Craft IIe
Apple IIe	Microdigital	TK-3000 IIe
Apple IIe	Spectrum	Microengenho II
MSX	Gradiente	Expert GPC-1
MSX	Sharp	Hotbit HB-8000
Sinclair Spectrum	Microdigital	TK-90X
Sinclair Spectrum	Timex	Timex 2000
Sinclair ZX-81	Apply	Apply 300
Sinclair ZX-81	Engebras	AS-1000
Sinclair ZX-81	Filcres	NEZ-8000
Sinclair ZX-81	Microdigital	TK-82C
Sinclair ZX-81	Microdigital	TK-83
Sinclair ZX-81	Microdigital	TK-85
Sinclair ZX-81	Prologica	CP-200
Sinclair ZX-81	Ritas	Ringo R-470
Sinclair ZX-81	Timex	Timex 1000
Sinclair ZX-81	Timex	Timex 1500
TRS-80 Mod. I	Dismac	D-8000
TRS-80 Mod. I	Dismac	D-8001/2
TRS-80 Mod. I	LNW	LNW-80
TRS-80 Mod. I	Video Genie	Video Genie I
TRS-80 Mod.III	Digitus	DGT-100
TRS-80 Mod.III	Digitus	DGT-1000
TRS-80 Mod.III	Kemitron	Naja 800
TRS-80 Mod.III	Prologica	CP-300
TRS-80 Mod.III	Prologica	CP-500
TRS-80 Mod.III	Sysdata	Sysdata III
TRS-80 Mod.III	Sysdata	Sysdata Jr.
TRS-80 Mod.IV	Multix	MX-Compacto
TRS-80 Mod.IV	Sysdata	Sysdata IV
TRS-Color	Codimex	CS-6508
TRS-Color	Dynacom	MX-1600
TRS-Color	LZ	Color 64
TRS-Color	Microdigital	TKS-800
TRS-Color	Prologica	CP-400

CLASSIFICAÇÃO DOS MICROCOMPUTADORES POR NOME

FABRICANTE	MODELO	PAÍS	LINHA
Appletronica	Thor 2010	Brasil	Apple II +
Apply	Apply 300	Brasil	Sinclair ZX-81
CCE	MC-4000 Exato	Brasil	Apple II +
CPA	Absolutus	Brasil	Apple II +
CPA	Polaris	Brasil	Apple II +
Codimex	CS-6508	Brasil	TRS-Color
Digitus	DGT-100	Brasil	TRS-80 Mod.III
Digitus	DGT-1000	Brasil	TRS-80 Mod.III
Digitus	DGT-AP	Brasil	Apple II +
Dismac	D-8000	Brasil	TRS-80 Mod. I
Dismac	D-8001/2	Brasil	TRS-80 Mod. I
Dismac	D-8100	Brasil	Apple II +
Dynacom	MX-1600	Brasil	TRS-Color
ENIAC	ENIAC II	Brasil	Apple II +
Engebras	AS-1000	Brasil	Sinclair ZX-81
Filcres	NEZ-8000	Brasil	Sinclair ZX-81
Franklin	Franklin	USA	Apple II +
Gradiente	Expert GPC1	Brasil	MSX
Houston	Houston AP	Brasil	Apple II +
Kemitron	Naja 800	Brasil	TRS-80 Mod.III
LNW	LNW-80	USA	TRS-80 Mod. I
LZ	Color 64	Brasil	TRS-Color
Magnex	DM II	Brasil	Apple II +
Maxitronica	MX-2001	Brasil	Apple II +
Maxitronica	MX-48	Brasil	Apple II +
Maxitronica	MX-64	Brasil	Apple II +
Maxitronica	Maxitronic I	Brasil	Apple II +
Microcraft	Craft II Plus	Brasil	Apple II +
Microcraft	Caft IIe	Brasil	Apple IIe
Microdigital	TK-3000 IIe	Brasil	Apple IIe
Microdigital	TK-82C	Brasil	Sinclair ZX-81
Microdigital	TK-83	Brasil	Sinclair ZX-81
Microdigital	TK-85	Brasil	Sinclair ZX-81
Microdigital	TK-90X	Brasil	Sinclair Spectrum
Microdigital	TKS-800	Brasil	TRS-Color
Milmar	Apple II Plus	Brasil	Apple II +
Milmar	Apple Master	Brasil	Apple II +
Milmar	Apple Senior	Brasil	Apple II +
Multix	MX-Compacto	Brasil	TRS-80 Mod.IV
Omega	MC-400	Brasil	Apple II +
Polymax	Maxxi	Brasil	Apple II +
Polymax	Poly Plus	Brasil	Apple II +
Prologica	CP-200	Brasil	Sinclair ZX-81
Prologica	CP-300	Brasil	TRS-80 Mod.III
Prologica	CP-400	Brasil	TRS-Color
Prologica	CP-500	Brasil	TRS-80 Mod.III
Ritas	Ringo R-470	Brasil	Sinclair ZX-81
Sharp	Hotbit HB-8000	Brasil	MSX
Spectrum	Microengenho I	Brasil	Apple II +
Spectrum	Microengenho II	Brasil	Apple IIe
Spectrum	Spectrum ed	Brasil	Apple II +
Suporte	Venus II	Brasil	Apple II +
Sycomig	SIC I	Brasil	Apple II +
Sysdata	Sysdata III	Brasil	TRS-80 Mod.III
Sysdata	Sysdata IV	Brasil	TRS-80 Mod.IV
Sysdata	Sysdata Jr.	Brasil	TRS-80 Mod.III
Timex	Timex 1000	USA	Sinclair ZX-81
Timex	Timex 1500	USA	Sinclair ZX-81
Timex	Timex 2000	USA	Sinclair Spectrum
Unitron	AP II	Brasil	Apple II +
Victor do Brasil	Elppa II Plus	Brasil	Apple II +
Victor do Brasil	Elppa Jr.	Brasil	Apple II +
Video Genie	Video Genie I	USA	TRS-80 Mod. I

UM LOGOTIPO PARA CADA MODELO DE COMPUTADOR

INPUT foi especialmente projetado para microcomputadores compatíveis com as sete principais linhas existentes no mercado.
Os blocos de textos e listagens de programas aplicados apenas a determinadas linhas de micros podem ser identificados por meio dos seguintes símbolos:

Sinclair ZX-81

TRS-80

TK-2000

MSX

Spectrum

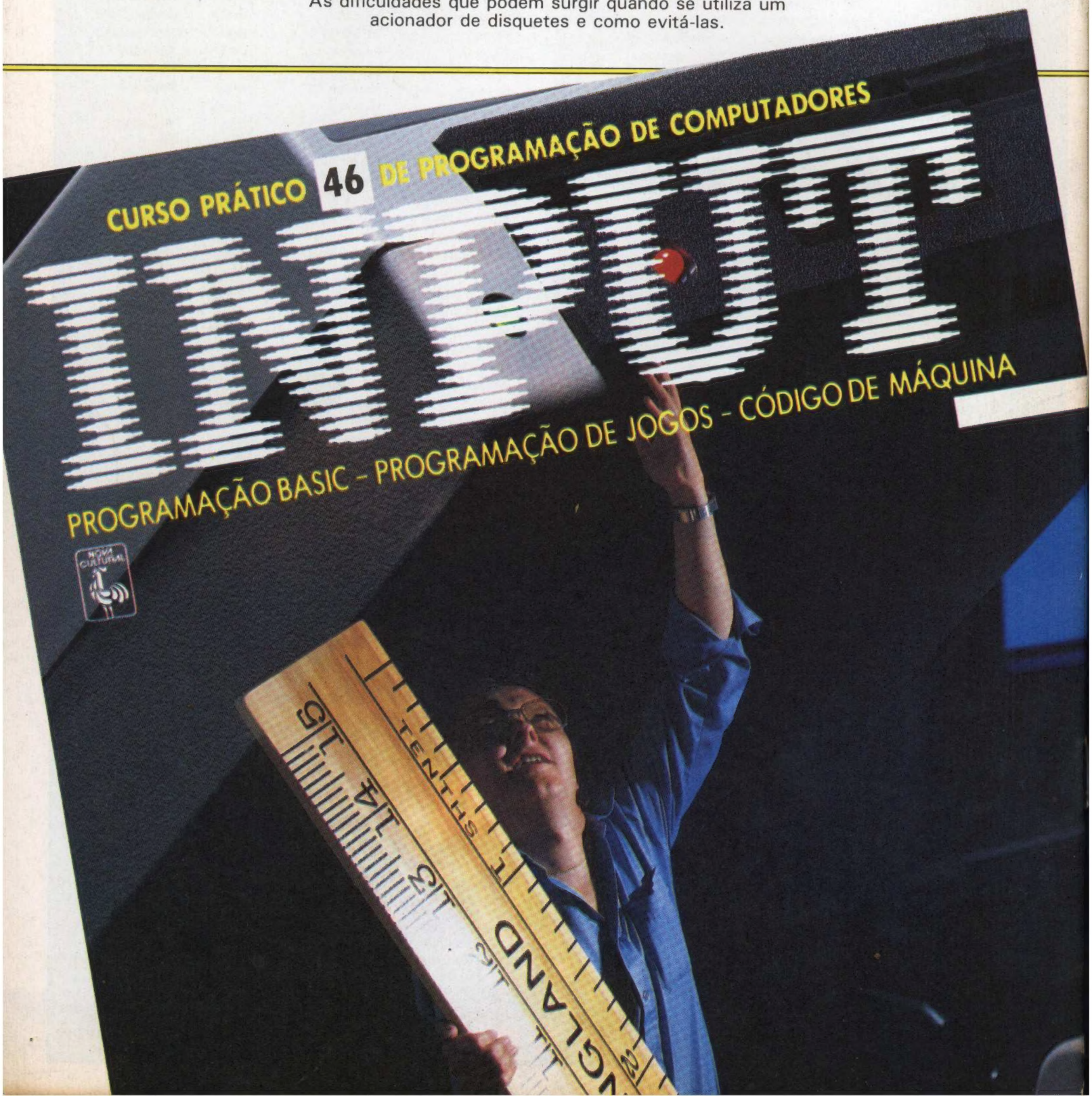
TRS-Color

Apple II

Quando o emblema for seguido de uma faixa, então tanto o texto como os programas que se seguem passam a ser específicos para a linha indicada.

NO PRÓXIMO NÚMERO

PROGRAMAÇÃO BASIC
Manchetes e letreiros. Como ampliar os caracteres da ROM. Montagem de letras garrafais com blocos gráficos.

PROGRAMAÇÃO DE JOGOS
Prossiga o jogo *A Raposa e os Gansos*, montando o programa e digitando as suas primeiras rotinas.

PERIFÉRICOS
As dificuldades que podem surgir quando se utiliza um acionador de disquetes e como evitá-las.